THE EVOLUTION OF COMPLEXITY IN AUTONOMOUS ROBOTS

A Dissertation Presented

by

Joshua E. Auerbach

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fullfillment of the Requirements
for the Degree of Doctor of Philosophy
Specializing in Computer Science

May, 2013

Accepted by the Faculty of the Graduate College, The University of Vermont, in partial fulfillment of the requirements for the degree of Doctor of Philosophy, specializing in Computer Science.

Dissertation Examination Committee:

Joshua C. Bongard, Ph.D. — Advisor

Margaret J. Eppstein, Ph.D.

Robert R. Snapp, Ph.D.

Charles J. Goodnight, Ph.D. — Chairperson

Domenico Grasso, Ph.D. — Dean, Graduate College

Date: February 26, 2013

Abstract

Evolutionary robotics–the use of evolutionary algorithms to automate the production of autonomous robots–has been an active area of research for two decades. However, previous work in this domain has been limited by the simplicity of the evolved robots and the task environments within which they are able to succeed. This dissertation aims to address these challenges by developing techniques for evolving more complex robots. Particular focus is given to methods which evolve not only the control policies of manually-designed robots, but instead evolve both the control policy and physical form of the robot. These techniques are presented along with their application to investigating previously unexplored relationships between the complexity of evolving robots and the task environments within which they evolve.

CITATIONS

Material from this dissertation has been published in the following form:

Auerbach, J. E. and J. C. Bongard. (2009). How robot morphology and training order affect the learning of multiple behaviors. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*.

Auerbach, J. E. and J. C. Bongard. (2009). Evolution of Functional Specialization in a Morphologically Homogeneous Robot. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Auerbach, J. E. and J. C. Bongard. (2010). Dynamic Resolution in the Co-Evolution of Morphology and Control. *Artificial Life XII: Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems*.

Auerbach, J. E. and J. C. Bongard. (2010). Evolving CPPNs to grow three-dimensional physical structures. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Auerbach, J. E. and J. C. Bongard. (2011). Evolving Complete Robots with CPPN-NEAT: The Utility of Recurrent Connections. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Auerbach, J. E. and J. C. Bongard. (2012). On the relationship between environmental and mechanical complexity in evolved robots. *Artificial Life XIII: Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems*.

Auerbach, J. E. and J. C. Bongard. (2012). On the relationship between environmental and morphological complexity in evolved robots. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

AND

Material from this dissertation will be submitted for publication to *PLoS Computational Biology* in the following form:

Auerbach, J. E. and J. C. Bongard. (2013). Environmental influence on the evolution of morphological complexity in machines.

in loving memory of

Alison Auerbach-Barr (1974-2003)

Acknowledgements

While a lot of my own hard work has gone into producing this dissertation, it would not have been possible without the vast support network–intellectual, emotional, financial, and logistical–that surrounds me.

I would first and foremost like to thank my advisor, Josh Bongard, for the innumerable hours he has spent with me hashing out new ideas, helping me interpret my results, editing papers, and discussing science. Josh has an extraordinary ability to state things in a way that both clarifies my own thinking and helps me to better convey my work to others. On many occasions I came to Josh feeling frustrated with some recent lack of progress, but his kind words would always make me feel better about my research, and motivate me to persevere. Moreover, I would like to thank Josh for the numerous opportunities that have been made available to me to attend workshops, conferences, and summer schools. Without his professional connections and the funding which he has secured these would not have been possible.

I would next like to thank the members of the Morphology, Evolution and Cognition Lab and the Vermont Complex Systems Center for many stimulating discussions which have helped to shape my research. I would also like to thank my committee for providing valuable feedback and asking tough questions, which have forced me to reconsider my assumptions, and have improved the quality of this dissertation. Additionally, I would like to thank the members of the evolutionary computation community, and in particular the GECCO Generative and Developmental Systems track for their inspiring research, fantastic feedback, and many stimulating conversations.

Next, there are a few people who have provided me an immense amount of logistical support, and deserve special recognition. Penni French of the Computer Science department has been an invaluable help throughout my time at UVM. Whenever I needed paperwork processed, a room reserved, or any other administrative task performed, Penni was ready to help, and in doing so made my life much easier. Likewise, Andi Elledge of the Vermont Complex Systems Center has been an immense help since I moved my office to Farrell Hall. I would also like to thank Jim Lawson of the Vermont Advanced Computing Core for all of the help he has provided me over the years: resolving issues and ensuring that I was able to run my experiments. If something was not working properly on the cluster, I always knew I could shoot an email to Jim and receive a prompt reply.

I am also fortunate to have many friends and family who have been incredibly supportive of me as I have pursued my graduate education. They have helped ground me, keep me sane, and have always been there when I needed someone to talk to. In particular I would like to thank my parents Ellen and Michael, and my brother Marc for all their love, support and inspiration.

Finally, I acknowledge that the work leading to this dissertation was supported by National Science Foundation Grant PECASE-0953837 and DARPA M3 grant W911NF-1-11-0076, and I also acknowledge the Vermont Advanced Computing Core which is supported by NASA (NNX 06AC88G), at the University of Vermont for providing High Performance Computing resources that have contributed to the research results reported within this dissertation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction and Literature Review

The ability to create more complex autonomous robots would be useful for a variety of reasons. Such robots could free people from repetitive tasks and perform actions that would be dangerous or impossible for humans to perform themselves. One approach to producing autonomous robots is evolutionary robotics, where the production of autonomous robots is automated through the use of evolutionary algorithms. This dissertation aims to address some of the challenges in evolving complex autonomous robots. Specifically, it focuses on methods that evolve both the control policy and physical form (morphology) of simulated machines. These techniques are presented and applied toward investigating previously unexplored relationships between morphological and environmental complexity.

This introductory chapter presents an overview of the published literature relevant to this dissertation. It primarily focuses on studies that incorporate aspects of robot morphology into evolutionary robotics. This includes the evolution of morphology alone, parameterized models of morphology and control, and topological evolution of complete robots. These works are compared along several different dimensions including the scope of evolution, the genome encodings utilized, the tasks investigated, the control policies employed, and transferability to physical robots. Additionally, relevant works on the evolution of complexity are discussed.

1.1 Introduction

It is now over a decade into the 21st century, and the vast majority of deployed robots are still constrained to operating only in structured environments such as factories. This is not because there is no use for robots elsewhere. On the contrary, robots that operate in outdoor or other unstructured environments such as the home or office would be of great social utility. So, what is preventing robots from making this migration from factories into our everyday lives? In order to operate in unstructured environments robots will need to be adaptive; that is, they must exhibit intelligent behavior.

What gives rise to such intelligent behavior? The principles of embodied artificial intelligence dictate that such intelligent behavior arises out of the coupled dynamics between an agent's body, brain and environment (Brooks 1999, Anderson 2003, Pfeifer and Bongard 2006, Beer 2008). One corollary of this concept is that the complexity of an agent's controller and morphology must match the complexity of the task or tasks that it is required to perform (Pfeifer and Scheier 1999, 455–466). However, when extending this idea to more complex agents in more complex environments, it is not clear how to distribute responsibility for different behaviors across the agent's controller and morphology. For example, if all a robot needs to do is follow a light source over flat terrain then wheels and a direct sensor-to-motor mapping would be an appropriate solution (Braitenberg 1986), but if the robot must be able to navigate over a variety of terrains and perform more complicated tasks then a more complex control strategy and morphology are required. This issue of scaling up complexity has been one of the major obstacles in developing robots capable of robust and adaptive behavior in unstructured environments.

On the other hand, biological organisms have evolved the capability to behave adaptively across an incredibly vast range of different environments. That this is a result of an evolutionary trend toward increasing complexity is often taken as fact (McShea 1991), but is a hotly debated topic among biologists and artificial life researchers (cf. (Bonner 1988, Gould 1996, Bedau 1998, Adami et al. 2000, Miconi 2008a, McShea and Brandon 2010)). Notwithstanding this debate, it is clear that evolution has produced a diversity of organisms possessing the types of intelligence that would be desirable to have in our robots. Moreover, robotics provides an experimental test-bed for investigating the relationships between morphological, neurological, and environmental complexity that would be difficult or impossible to perform with biological organisms. This dissertation describes such an investigation.

1.2 Evolutionary Robotics

Evolutionary robotics (Harvey et al. 1997, Nolfi and Floreano 2000), a biologically inspired technique in which evolutionary algorithms are employed to optimize the control policy of a robot, has provided one framework for overcoming the limitations of human intuition in designing robust, non-linear, control strategies for autonomous machines. However, the majority of work in evolutionary robotics has done only that: optimize control strategies for a human designed or biologically inspired robot morphology. This methodology has severe limitations: fixing a robot's morphology places limits and biases on the kinds of actions that the robot can perform, and therefore also on the more complex behaviors that those actions may eventually support. For example a wheeled robot with a rigid gripper can only move over relatively even terrain and grasp objects of fixed dimension.

However, there are ways to overcome these limitations. Evolutionary algorithms may be used to optimize a robot's physical form and structure (also known as its morphology) in addition to its control policy. Sims (1994b) first introduced an evolutionary framework in which both the morphology and control of simulated machines were optimized in virtual environments to produce adaptive behavior. This work has been followed by other studies (see below) in which aspects of the machine's morphology and control were evolved in virtual environments. This approach has the advantage of discovering body plans and sensor configurations appropriate for the machine's task environment rather than being artifacts of human design biases or copies of animal body plans only appropriate for that animal's ecological niche.

This brings up a fundamental difference between evolutionary algorithms and more formal optimization methods such as learning (Sutton and Barto 1999, Abbeel and Ng 2004, Tassa et al. 2008): the latter are suited for parametric optimization requiring guarantees of convergence, while the former allow for topological improvement where such guarantees cannot be made. The majority of currently popular optimization techniques rely on having a fixed number of parameters that must be optimized: these parameters may specify the control policy of a robot, or in some cases also aspects of its morphology (Dollar and Howe 2007). However, several evolutionary algorithms exist that do not assume a fixed structure exists whose parameters must be optimized, but rather that structures and their parameters may be improved simultaneously (Koza 1992, Bongard and Lipson 2007).

This chapter discusses and compares the various published works in which evolutionary algorithms have been used to optimize some or all aspects of a robot's[1] morphology in addition to its control policy. There are numerous dimensions by which this work may be classified, including but not limited to: (1) how much of control and morphological structure was placed under evolutionary control; (2) the tasks that robots were evolved to perform; (3) the sorts of genotypic encodings employed; (4) whether experiments were carried out entirely in simulation or whether physical robots were constructed or otherwise utilized; (5) whether sensory systems were included such that controllers received feedback from their environments; and (6) the sorts of control policies used. The first of these dimensions has been chosen to determine the high level structure of this chapter, though all dimensions will be discussed. The following section presents several experiments in which only morphology was under evolutionary control (either for unactuated objects or for robots with a predefined control policy). After this, experiments in which morphology and control were jointly evolved are discussed, starting with examples in which some parameters of the morphology were evolved and then moving on to experiments in which the topology of the morphology and controller were evolved, thus allowing for the creation of robots of arbitrary shapes and behaviors. Following these discussions, a brief overview of some of the questions surrounding the evolution of complexity are discussed, along with how evolutionary robotics has been able to contribute to this debate. Finally, this chapter ends with some conclusions that may be drawn from these various studies along with a brief outline of the remainder of this dissertation.

1.3 Evolution of Morphology

There have been several publications detailing work in which evolutionary algorithms were employed in the generation of three-dimensional physical structures. Often evolving structure alone is a first step in a research trajectory eventually geared toward joint evolution of robot morphology and control (cf. (Eggenberger 1997, Funes and Pollack 1997, Parker et al. 2003, Auerbach and Bongard 2010b)). This work will be described in the following paragraphs. Other work has looked at evolving physical structures themselves as an end goal either for engineering or artistic purposes (cf. (Bailly-Salins and Luga 2007, Clune and Lipson 2011)). While this second class is related and may make use of many of the same techniques used for evolving robot morphologies, the goals of these works are different and so they will only briefly be touched upon here.

[1] When evolving robots in simulation as is the case with majority of work discussed in this chapter *virtual robots* are often referred to as [virtual] *creatures* or *agents*. All three of these terms will be used interchangeably throughout this chapter.

Funes and Pollack (1997) evolved static LEGO structures in simulation, which were then then physically constructed. The simulator modeled connections between LEGO pieces as rotational joints so that a collection of multiple connected pieces was modeled as a network of joints with different capacities and external forces. This network was required to be in static equilibrium for the structure to not collapse. A direct tree based genomic representation was used such that each LEGO brick was represented with its size type and a list of its descendants including their position of connection, rotation and size type. Using a standard steady-state genetic algorithm (GA)[2] (Holland 1975) the researchers were able to evolve bridges, scaffolds, supportive crane arms, and tables, all of which were then physically assembled and tested with real LEGO bricks. Using a predefined set of components so that results may be physically fabricated is a popular approach in this domain, more examples of which will be presented in later sections.

Relevant work in this area was also performed by Eggenberger (1997) around the same time. In this work he examined evolving 3-D shapes for bilateral symmetry using a developmental encoding based on differential gene expression. This was a model heavily influenced by biology and incorporated gene regulation, cell division and death, cell differentiation, and positional information. Specifically, a structured genome was employed that contained two classes of genes: regulatory units and structural genes, where several regulatory units could determine the activity of one or several structural genes. Starting with a single cell, a structure was grown based on cells' affinity for a variety of transcription factors, cell adhesion molecules, and receptors. This, combined with the presence of "chemical" morphogens in the environment allowed cells to differentiate and form desirable structures (in this case structures were evolved that possessed bilateral symmetry). Other related biologically-inspired models will be discussed towards the end of this chapter.

Hornby and Pollack

Hornby and Pollack demonstrated the usefulness of generative grammatical encodings for designing three-dimensional physical objects by evolving a class of formal grammar known as Lindenmayer Systems (L-Systems) to produce tables (Hornby and Pollack 2001a). Later they extended this methodology to evolving morphologies and controllers of 3-D robots (to be discussed later), but the majority of their methods will be described here.

[2] A genetic algorithm (GA) is a form of stochastic optimization based on the process of natural selection. It is one of the most common forms of evolutionary computation (EC).

L-systems are an abstraction of biological development composed of grammatical rewrite rules (Lindenmayer 1968). An L-system begins with an initial variable and applies rewrite rules in parallel to all eligible variables in a string, similar to how cells divide in parallel in biological organisms. Each rule is of the form $A- > Bb$ where A, B are variables (and so appear on the left hand side of some rule) and b is a constant (not appearing on the left hand side of any rule). In the case of Hornby and Pollack's work, the constants were all commands which guided the movements of a LOGO-style turtle through a three-dimensional space of voxels. All voxels began empty, but were filled as the turtle entered them.

Specifically, Hornby and Pollack employed a more general class of L-system: parametric, context-free L-systems (P0L-systems). P0L-systems allow for production parameters to be associated with each variable. The right hand side of the rewrite rules become conditional expressions which depend on the value of this parameter. In their work, conditions were restricted to being comparisons as to whether a production parameter was greater than a constant value. Besides commands which direct the movement of the turtle, their design language allowed for two special operations: enclosing square brackets ('[', ']') pushed and popped the current state of the turtle (position and orientation) to and from a stack, and *block*(n) repeated the enclosed block n times.

Starting with a population of randomly generated L-systems, Hornby and Pollack evolved tables through the use of specially-designed genetic operators (both sexual and asexual). Their fitness function consisted of several terms which were multiplied together to obtain a total fitness. The terms selected for maximizing height, surface structure, stability, and minimizing the number of excess voxels to prevent the formation of solid volumes. They compared this method to one in which the production commands were evolved directly and the block replication operator was not allowed. The tables produced by the L-systems were much more likely to produce good structures and include regularities in addition to receiving significantly higher fitness scores. Further experiments studying the inclusion of block replication with direct evolution and exclusion of block replication with evolving grammars demonstrated that both of these methods performed better than not having replication or rewrite rules, but not as good as including both mechanisms.

This work provided solid evidence for why a generative encoding should be used. One reason for this is that modules may be repeated. As an example of this consider changing the height of a table. With a generative encoding it is likely that each leg is a repetition of the same genetic material and so the height can be altered by mutating a single underlying rule. On the other hand, if a direct encoding is employed, adjust-

ing the height while maintaining a flat surface would require mutating each leg independently, an unlikely occurrence. Additionally, as shown by example in this work, changing the replication parameter from four to three easily changed a grammar from producing four legged tables to one that produced three legged tables. This work is of further importance because the authors were able to fabricate their evolved tables using rapid prototyping (3-D printer) technology.

Related work by Paul et al. (2005) and Rieffel et al. (2009) used a direct encoding and L-systems respectively to evolve tensegrity structures which could then be fabricated as physical objects using rapid prototyping technology. Tensegrity structures are self-supporting structures composed of rigid struts connected by cables (Fuller 1961, Pugh 1976). This work is of interest because it provided additional evidence for the benefits of generative encodings, and because tensegrity structures may be actuated to form tensegrity robots such as those described in (Paul 2006) and (Rieffel et al. 2010).

Komosinski and Rotaru-Varga

Similar to Hornby and Pollack, Komosinski and Rotaru-Varga (2002) explored how different genome encodings affected the performance of their "Framsticks" evolution system on a variety of tasks. The first of these tasks–maximizing the height of stable static structures (with physics modeled using a finite element method)–will be discussed here, while the subsequent tasks will be discussed in later sections. They evolved virtual creatures composed of sticks which were built of two material endpoints connected through a flexible rod. When evolved for dynamical behavior, sticks were actuated at shared endpoints with three mechanical degrees of freedom, but these degrees of freedom remained fixed for the creation of static structures as discussed here.

Three different genomic encodings were investigated. The most simple of these was a direct representation which consisted of a list of all components and attributes, denoted as **simul**. Other representations were translated into a **simul** representation before being constructed. The second encoding was a direct recurrent encoding denoted **recur**, and was defined by a nested list. This list defined the presence and connections of sticks. In this encoding, modifiers to attributes were also included that affected not only the following stick, but subsequent sticks in the genotype with decreasing weight. Finally an indirect developmental encoding **devel** was also explored. This encoding involved passing through a developmental phase. The representation

was similar to **recur**, but instead of being constructed offline by a builder, the genome was interpreted by growing cells themselves, and allowed for cell division and repetition of modules.

When these three encoding were used to evolve static structures for maximum height, the average best fitness values increased going from **simul** to **recur** to **devel**, that is in conjunction with increasing the level of abstraction of the underlying representation. While all three encodings produced a variety of solutions more regularity was seen with the **devel** encoding. This also provided further evidence for why one may want to employ a generative or developmental encoding. However this was a simple task and thus may not be as meaningful to evolving complete robots as their subsequent experiments involving actuation, which will be discussed in later sections.

Parker et el

Similar to Funes and Pollack, Parker et al. (2003) also evolved static LEGO structures. They evolved towers of LEGO pieces (from a predefined pool) for height and stability in simulation. Their physics simulation evaluated if a structure would break due to gravity. If that was the case then the fractured part was removed and the structure was re-evaluated. This process continued until a rigid stable structure was found. A genome which directly encoded the pieces in the structure along with their connections was employed. This genome was of variable length, thus allowing for arbitrarily sized structures. Structures were evaluated based on their stability, tension optimality, height, and weight. Through these methods the authors were able to evolve tall towers with structural integrity and stability.

Later, Parker et al. (2007) extended this work to evolve LEGO robot body plans for wheeled locomotion using fixed controllers that chose optimal directions of movement. Morphologies were once again assembled from a collection of LEGO parts, except in this case the assembly was required to include one of the control bricks so that the structure could be powered. If an evolved morphology had a wheel touching the ground and attached to a motor then it could move, otherwise it was deemed static. Based on the alignment of such active wheels, a control program determined the optimal direction of movement to reduce friction. From this direction of movement the ability of the morphology to move was calculated. Some special adjustments were necessary to obtain stable movement in early generations due to the unlikeliness of evolution creating two counter-balanced wheels, but evolution was then able to maximize movement and stability in order to find wheeled robots.

Hiller and Lipson

Other relevant work was recently carried out by Hiller and Lipson (2010). This work was extremely novel in that instead of evolving the morphologies of rigid structures or robots the researchers evolved the morphologies of volumetrically actuated[3] amorphous (soft) robots for locomotion. In this work, three different representations were explored for defining the presence of one of several materials: discrete cosine transforms (DCTs) (a form of Fourier transform), compositional pattern producing networks (CPPNs) (Stanley 2007)[4], and Gaussian mixture models. The different materials had different thermal expansion coefficients which allowed for actuation by applying a temperature oscillation. Originally these experiments were carried out with a custom designed simulator, and in a later publication (Hiller and Lipson 2012) they were physically fabricated and actuated through the use of a pneumatic chamber.

Others

Additional related work was performed by Bailly-Salins and Luga (2007). They presented a system for the generation of artistic 3-D shapes. However, in this case, in lieu of an objective function an interactive evolutionary system was implemented in which users decided fitnesses of objects based on their subjective preferences. The system was based on a two-stage genetic programming (GP)[5] approach (Koza 1992) in which one stage of GP acted on potential functions that algebraically defined shapes, and the second stage acted on operators combining these evolved shapes into more complex objects. Users were able to modify either the underlying shapes directly or compositions of these shape to produce desirable 3-D objects. This work demonstrated how evolution of structure alone could be useful in and of itself for producing desirable and useful objects (e.g. seats).

More recent work by Clune and Lipson (2011) has also employed interactive evolution towards generating artistic 3-D structures. Their Endless Forms website (`http://www.endlessforms.com`) allows users on the world wide web to evolve their own 3-D structures (encoded by CPPNs) either from scratch or by starting from a structure evolved by others. Users are shown the current population of structures and allowed to select their favorites for reproductions, similar to Dawkins' Blind Watchmaker (1986). Their web interface also has a tie-in with a web-based 3-D printing service: `http://www.shapeways.com`, allowing for

[3] These robots are actuated through the use of materials capable of controlled expansion and contraction.
[4] CPPNs are a recently introduced abstraction of development which will be discussed in detail later on in this dissertation.
[5] Genetic programming (GP) is an evolutionary algorithm (EA) where genomes are computer programs or mathematical functions.

the evolved structures to be 3-D printed and delivered to a user. Their website is similar to `http://www.picbreeder.org` (Secretan et al. 2011), an older website which also uses interactive evolution and CPPN genomes for evolving images.

Interactive evolution was also employed by Cussat-Blanc and Pollack (2012) for evolving the morphologies of modular robots. They employed a developmental model based on genetic regulatory networks (GRNs) for constructing modular robot morphologies composed of cuboid blocks which could divide and differentiate. The resulting morphologies were allowed to contain three types of blocks: *noop* which were used to build structures and could be linked to other blocks, but otherwise had no function. *Hinge* blocks were allowed to rotate through a central axis within a predefined range. *Motor* blocks, which actuated the hinge blocks similarly to how muscles actuate joints in biological organisms. These evolved morphologies were coupled with pre-defined oscillators and placed in a physics simulator[6]. A set of nine morphologies was concurrently presented to a user (in the same environment) and the user evolved the morphologies through interactive evolution. However, unlike Endless Forms or Picbreeder it does not appear that this interactive system was made widely available on the world wide web, and so all interactive evolution was carried out by a single user in isolation.

While human-guided evolution is not the focus of this chapter, these ideas may be employed toward producing novel robots. This technique has also been taken by Lund and Miglino (1998) and Miglino et al. (2007). Oftentimes it is difficult to create a fitness function that will produce robots with a desired behavior, but by incorporating interactive evolution it may be possible to leverage human intuition towards evolving more successful robots.

1.4 Evolution of Morphology and Control

The main objective of this chapter is to present an overview of research done to date in which both the morphology and control of robots or simulated robots were jointly evolved. This sections describes research in which this approach has been taken.

[6]In this work the Bullet simulation engine: `http://www.bulletphysics.com/` was used.

1.4.1 Parametric Evolution

One possible method of jointly evolving morphology and control is to parameterize certain properties of a robot, and place these parameters under the control of an evolutionary algorithm. This subsection covers a diverse set of research projects in which this approach was taken.

Balakrishnan and Honavar

The most common approach taken to parameterize aspects of the robot's morphology is to enable an evolutionary algorithm to alter aspects of the robot's sensory system. One early piece of work in which this was accomplished was presented by Balakrishnan and Honavar (1996). In this work the authors evolved sensor placements and operating ranges for a simulated Khepera-like (Mondada et al. 1994) robot acting in a two-dimensional gridded arena. The environment contained "boxes" which the robot was tasked with clearing to the edges of the arena. The number of sensors was set to 8, but evolution was allowed to decide the placement of the sensors around the robot as well as determine the range of each sensor (up to a maximum range) and finally could, with low-probability, turn off sensors, although no penalty was imposed for having a greater number of sensors. These parameters were evolved along with the connection weights for a mostly fixed-topology neural network. The neural network had a predefined feed-forward topology, but with a low probability recurrent connections could be added. One result of this work was that even without a penalty for extra sensors the most fit individuals did indeed switch off some the sensors. This demonstrated that a reduced sensor set could work better than the complete set, which could reduce cost and improve performance if used to guide the construction of physical robots.

Mark et al.

Related work was carried out by Mark et al. (1998). Here, once again, a simulated two-dimensional environment was utilized. The simulated robots employed were similar to Braitenberg Vehicles (Braitenberg 1986). The robots had circular bodies, primitive vision sensors ("eyes") and were controlled by two wheels. In addition to several types of neural controllers, both the width (angle of view) and number of eyes (uniformly distributed around the robot) were evolved for two different tasks. The first of these tasks involved placing a number of these robots into an environment, each with an initial energy. The robots then interacted with each other, with obstacles, and with lamps in the environment. Energy was consumed during a robot's

lifetime. A robot 'died' when its energy fell below a threshold, but energy could be gained by dwelling in the light of a lamp. Additionally, a greater quantity of energy was lost when two robots collided. Meanwhile, colliding with an obstacle resulted in 'instant death'. A second task was also investigated in which a single robot had to navigate to a target while avoiding obstacles. Incremental evolution was used such that at first few obstacles were present, but as successful solutions were found an increasing number of obstacles were added. While the results presented were inconclusive (the authors noted that typical runs lasted between one to two weeks with the computational resources available at that time, meaning the ability to make statistical statements was severely limited), this work did explore many promising techniques which have since been examined in greater depth by other researchers. These included: sensor evolution, competing robots, incremental evolution, and speciation (robots were placed in families and sexual reproduction was only allowed between members of the same family).

Lichtensteiger and Eggenberger

Around this same time, Lichtensteiger and Eggenberger (1999) evolved the relative positions of light sensors on a physical robot. A specially-designed robot was constructed with 16 light sensors attached to its motors such that each light sensor could be rotated independently. The angular positions of each sensor were evolved directly by an evolution strategy (ES) (Rechenberg 1973). A motion parallax task was investigated using a fixed homogeneous neural network controller, in which the robot was tasked with avoiding an obstacle if its lateral distance was closer than a critical value, and not avoiding it otherwise. Carrying out this sort of evolution on a physical robot was quite novel for the time, but also necessitated many constraints. For example, the robot moved on a track at fixed velocities, and the neural network controller was only used for determining whether or not the robot would have chosen to avoid the 'obstacle' placed in the environment. Additionally, evolution only ever altered the position of one sensor at a time, which then became fixed as evolution moved on to the next sensor.

Bugajska and Schultz

Another piece of work in which sensor parameters and placement were evolved, this time for micro air vehicles, was presented by Bugajska and Schultz (2000). As opposed to neural controllers, their flying robots were controlled by evolved stimuli-response rules. Micro air vehicles were evolved to navigate to a target

location while avoiding obstacles (trees, modeled as cylinders topped with spheres) in a flight simulator. Two off-the-shelf GA systems–SAMUEL (Grefenstette 1991) and GENESIS (Grefenstette 1984)–were used. SAMUEL evolved the stimuli-response rules while GENESIS evolved the properties of the sensors: number of sensors and sensor beam width. This worked by allowing SAMUEL to run for 60 generations for every potential sensor configuration in the GENESIS population. The goal of this work was to minimize the weight and power requirements of the vehicle and so having fewer sensors was preferred. Only preliminary results were presented, so it is difficult to gauge the success of this approach, but it was novel to apply GAs to the evolution of autonomous flying vehicles.

Bauson et al.

Buason et al. (2005) evolved sensor parameters (range and angle of camera) in a simulated co-evolutionary predator/prey situation along with the weights of fixed topology neural network controllers. The robots were simulated Kheperas existing in the YAKS simulator[7]. The authors investigated how allowing either the predator, prey, or both to have the ability to adapt these sensor parameters affected evolution, and also explored how tying a maximum speed to view angle altered this. They showed that when just evolving sensor parameters predators preferred a small view angle with a long range, while prey evolved wide view angles with shorter ranges. Additionally, when there was a trade-off between speed and view angle, prey would choose speed over being able to see.

Parker and Nathan

Another study by Parker and Nathan (2007) examined the evolution of sensors for a hexapod robot. The authors worked with the ServoBot: a hexapod with two degrees of freedom per leg and equipped with a sensor base on which resided four tactile sensors, four infrared sensors, and four ultraviolet sensors. All sensors were binary: they either detected a stimulus or they did not. The sensors were in fixed positions, but a GA was responsible for evolving which sensors were active, the orientation of all the sensors, and the range of the infrared and ultraviolet sensors. The robot was simulated in a 300cm x 300cm arena and tasked with traversing from a variable start position to a fixed target position at which resided an omnidirectional ultravi-

[7] `http://freecode.com/projects/yaks`

olet light source. The arena additionally contained obstacles configured in seven different configurations and a fixed position omnidirectional infrared light source.

A form of control policy was also under the control of the GA. Sixteen predefined gaits were made available. The GA dictated the consequents of 13 rules of the form "if sensor detects a stimulus then trigger gait number x". The 13 rules defined one gait for each sensor plus a default gait. In all of the authors' experiments (except for one environment that was too difficult) the simulated robots were able to evolve successful strategies to reach the target. These most commonly involved a wall-following behavior plus approaching the target directly when there was a direct path. One unexpected result was that even though IR light was present in the environment, all evolved agents switched the IR sensors off, because they could see the UV light over the obstacles. As the authors stated, this could save considerable expense when equipping the physical robots with sensors.

Other Morphological Parameterizations

Besides the sensors of a robot it is also possible to parameterize other aspects of a robot's morphology such as body segment sizes, joint ranges, wheel radii, and so on. Several publications in which researchers have taken this approach are described below.

Lee et al.

The first publication of this sort was contributed by Lee et al. (1996)[8]. They investigated evolving obstacle avoidance on a simulated, circular, two-wheeled robot (much like the Khepera) (see also Lund et al. (1997)). They evolved control programs using GP. These control programs were boolean trees which took in values from infrared sensors and computed three binary outputs dictating wheel direction and speed. The sensors were evolved as terminal nodes and had an associated value dictating their relative angle to the robot's heading direction. At the same time a GA was used to evolve a vector of real numbers representing parameters of the robot's morphology: motor time constant, size of wheel base, wheel radius, and body size. They were able to successfully evolve simulated robots that could move around while completely avoiding obstacles in a number of simulated environments. Additional experiments took an evolved controller and put it in other

[8]Other work that may fit into this class was carried out by Ventrella (Ventrella 1994) who evolved several parameters of simulated stick figures, however his approach was that of a visual artist primarily interested in animation, not in designing robots.

morphologies where it was shown that performance seriously degrades when using body plans different from the one optimized with the control strategy.

More recent work by Lund (2003) has evolved parameterized weights for a fixed topology neural network controller and morphological parameters for LEGO robots in simulation which were then built with real LEGO MINDSTORM kits. Specifically, the evolutionary algorithm could choose from three different kinds of LEGO wheel (having different diameter, width, friction, and so on), 25 different wheel bases on a LEGO axis at the robot's center, and 11 different positions of two LEGO light sensors (dictated by the possible attachment points on the LEGO stud). These robots were evolved to perform line following tasks of increasing complexity. A form of minimal simulation (defined by Jakobi as "the simplest type of simulation capable of evolving controllers for real robots" (Jakobi et al. 1998)) was combined with a lookup table of recorded behavior of the real components. Variability in evolved morphologies was observed. Specifically, distance between sensors increased with increasing line width, and wheel base diameter decreased with increasing line curvature (allowing for sharper turns).

Endo et al.

Around this same time Endo et al. (2002) evolved controllers composed of neural oscillators with predefined structure and aspects of morphology for a simulated humanoid robot modeled on the open-source PINO robot. The fitness function rewarded for distance traveled in a given amount of time. This robot was composed of several links making up the torso and legs. In their first experimental step relative links lengths were evolved while keeping the total length of all links constant. In a second experimental step aspects of the controlling servomotors such as the gain of the proportional-derivative (PD) controller, maximum torque, inertia, and others were also evolved. Allowing the details of the servomotors to be modified by evolution in this way was a novel contribution and may be useful when creating the motors for physical robots in the future.

More recently there have been several pieces of research applying similar ideas to the evolution of legged robots. Wampler and Popović (2009) used a hybrid optimization method with an outer loop based on the covariance matrix adaptation evolution strategy (CMA-ES) (Hansen and Ostermeier 2001), and a derivative based space-time optimizer as the inner loop, to evolve gaits and limb sizes for a variety of animal-like

creatures. Their objective function selected for efficient locomotion without "wiggles" while maintaining head stability. These properties are seen in natural organisms, and have come about due to selection pressures (e.g. head stability simplifying vision/optical flow) not directly simulated. With these methods they were able to realize a variety of gaits seen in natural animals as well as finding gaits for non-existent (e.g. monopeds, pentapeds) and extinct animals (e.g. a velociraptor).

Heinen and Osório (2009) evolved two types of stable locomotion controllers for a simulated quadruped based on a medium sized dog: recurrent artificial neural networks (ANNs) of fixed topology and finite state machines (FSMs). The ANNs were shown to take longer to evolve due to the larger search space, but achieved significantly better fitness in the long run when compared with the FSMs. The authors then presented additional experiments in which aspects of the morphology (including what appeared to be the size of components) were also evolved. Their results showed that robots with evolved morphologies significantly outperformed those with the predefined morphology. These results lend further evidence of how allowing evolution to control aspects of morphology in addition to control may be beneficial.

Bongard and Paul (2001) investigated parameterizing morphological aspects of a simulated[9] bipedal robot tasked with locomoting. Specifically, they used a simulated biped with five links (a waist, two upper legs, and two lower legs), and predefined haptic and proprioceptive sensors controlled by neural networks. They compared the robot as designed (without morphological parameterizations) to robots which had the length and radii of the legs and waist under evolutionary control. Additionally, they compared experiments in which each link had an associated mass block on it with possibly evolved sizes and positions. In all of these experiments the variants with evolvable morphological parameters repeatedly reached higher fitnesses suggesting that even though the search space had been expanded it was a useful expansion. At the same time, all the runs in which the robots had attached mass blocks on their links did not reach the performance of those without mass blocks, suggesting that arbitrarily increasing the number of morphological parameters is not necessarily useful.

Another example of parameterized morphological evolution was given by Römmerman et al. (2009). They evolved parameters of morphology and control for a hexapod robot to be used for exploring craters on the moon or other planets. Properties of the morphology such as limb length and placement were evolved along with control parameters for an inverse kinematics controller based on walking patterns. Evolution

[9]Simulations conducted with MathEngine Dynamic Toolkit from MathEngine PLC, `http://www.mathengine.com`.

was carried out with CMA-ES and each candidate solution was evaluated in three environments: one with a positive inclination, one with a negative inclination, and flat ground corresponding to the walls and bottom of a crater that the robot would need to explore. Additionally, the flat ground environment used various different friction coefficients so that resulting behavior was not limited to a single type of friction, but robust to many surfaces. Several good solutions were achieved with different combinations of morphology and walking patterns. Having such a diverse set of possible solutions should allow for greater flexibility when building a physical robot that will perform the desired task.

A few more recent instances of parameterized approaches to evolving morphologies can be found in (Moore and McKinley 2012) and (Clark et al. 2012). In the first of these publications the researchers evolved parameters such as arm length and foot radius, as well as flexibility properties of a passive joint for a robot modeled on *Oxudercinae* or mudskipper. Here, the movement was controlled by a predefined oscillator. By evolving for all of the possible parameters the robots were able to find solutions that maximized their ground contact area in a low friction environment. In the second of these publications the authors evolved the shape (rectangular dimensions) and stiffness of the caudal fin of a swimming robot. These evolved parameters were then transferred to a physical robot, and a correlation between evolved results and actual behavior was demonstrated. Techniques such as these are useful when constraints dictate that a robot must take on a particular form, but leave open the possibility of optimizing component dimensions and the material properties of those components.

1.4.2 Topological Evolution

While many of the research projects just discussed demonstrated the benefit of including morphology in the scope of variables subject to evolutionary control, they all had severe limitations in that they could only find solutions within the parameterizations designed by a human researcher. However, as discussed in the introduction, one of the key benefits of evolutionary computation over other optimization techniques is the ability to explore more than just a parameterization of a given model. Specifically, it is possible to evolve not only the parameters of a robot's morphology and control architecture, but to explore the much wider space of the topology of the morphology and control architecture. This subsection presents work in which this approach is taken, starting with the work of Karl Sims and those employing similar techniques, and then moving on to other forms of encodings and developmental approaches.

Sims' Virtual Creatures

The most well known and often cited work in which the topology of morphology and control of virtual robots were evolved at the same time was performed by Karl Sims over a decade and a half ago. In a seminal article Sims (1994b) presented a system in which virtual creatures (robots) were evolved for a variety of behaviors including swimming, walking, jumping and phototaxis. In subsequent work (Sims 1994a) pairs of competing robots were evaluated at the same time to judge their competency at a box grabbing competition. Being the most well known work in this domain as well as the inspiration for much of the other work discussed in this chapter, Sims' work will be described in greater detail than others.

Sims' creatures were composed of a collection of rigid rectangular solid (cuboid) body segments connected by various joint types (rigid, revolute, twist, universal, bend-twist, twist-bend, or spherical). The joints were actuated by effectors which simulated muscle forces as dictated by output neurons in an embedded neural network controller. Additionally, the neural network was provided with input from various sensors contained within different body segments. Sims provided for three types of sensors: joint angle sensors provided proprioception by giving the current angle for each degree of freedom of each joint; contact sensors provided tactile input and produced a binary output depending on whether the sensitive region within a given segment was in contact with another object; photosensors provided directional information relative to global light sources. Unlike traditional neural networks in which each neuron computes a threshold or sigmoid function on a weighted sum of inputs, Sims' neurons could perform diverse functions on their inputs. These functions could be strictly dependent on their inputs or could produce time-dependent outputs.

A genetic algorithm was used to evolve creatures for a particular task. Evolution acted on a genetic representation that was used to produce the phenotypes described in the previous paragraph. The genetic representation employed by Sims was a form of nested directed graph. The top-level graph represented the creature's morphology. The phenotype of cuboid parts was made from this graph by starting at a designated root node and tracing through the connections of the graph. Recurrent and cyclic connections were allowed as well as multiple connections between pairs of nodes. Including these abilities made the encoding capable of representing structures with repeated modules and fractal-like structures.

While connections between nodes define the connectivity of body segments, additionally each node contained internal properties describing the segments produced from it. These properties included the dimensions of the cuboid; *joint-typ*, which specified which of the joint-types was used to connect the given node to its par-

ent; *joint-limits*, which constrained the movement of the specified joint; and, finally *recursive-limit* specified how many segments were to be made from this node if it was part of a cycle. Additionally, each connection in the graph contained properties which represented the placement and scaling of a child part relative to its parent: *position, orientation, scale,* and *reflection* (which allowed for generating symmetric creatures). Finally a *terminal-only* flag on a connection could cause a connection to be applied only after a recursive limit was reached so that, for example, a hand-like appendage could be attached to the end of a repeating chain of units.

The neural network control architecture was also represented as a directed graph. Each node in the top level (morphological) graph contained a graph of the neural nodes and connections present within it. This allowed for similar morphological units to be controlled by similar but independent control systems. Neural connections were allowed within a given physical part as well as between neighboring parts and between any neuron and a central set of neurons that was not associated with any part of the morphology.

Once a given morphology (and embedded control network) was created from a genotype it was placed in a physical simulation to evaluate its fitness on a particular task. Sims used a custom-designed physical simulator which included dynamics, collision detection and response, friction, and an optional viscous fluid effect. Swimming, land locomotion, jumping and light following creatures were all evolved based on their performance in a given environment in the physical simulator. The GA, including fitness evaluations in the physical simulator, were all carried out in parallel on a Connection Machine® CM-5 in a master-slave message passing model.

Since the graph-based genotype was a non-standard representation for a GA, Sims needed to design custom genetic operators. Mutations were accomplished via a five step process. First, internal node parameters were subjected to possible alterations in which, based on a defined frequency, boolean flags were flipped and scalars were modified by adjustments from a Gaussian distribution, with scale corresponding to the original value. Possible negations were also allowed, and after alteration values were clipped to being within a predefined valid range. Second, a new random node was always added (but would be thrown out if no new connections formed to it). Third, internal connection parameters were mutated in the same way as the node parameters, with the added chance of a connection being moved to point to a new node with a given probability. Fourth, new random connections were added and existing connections removed. Finally any element not reachable from the root node was garbage collected to prevent unnecessary growth in the size of the graph.

Sexual reproduction was also implemented via two different mating methods. The first was crossover where nodes were aligned in the way they were stored and nodes (including connections) from the first parent were copied to the child, except that at one or more crossover points the copying source was switched to the second parent. The second method was grafting. Here the first parent was copied, except one of its connections was chosen to be randomly reassigned to point to a random node in the second parent which, along with its descendants, was added to the child. Any unconnected nodes from the first parent were discarded.

Through these methods, Sims was able to successfully evolve virtual robots from an initially random population for all of the tasks which he investigated. A video of some of the evolved creatures is available from the internet archive at `http://www.archive.org/details/sims_evolved_virtual_creatures_1994`.

Descendants of Sims

Since Sims' initial publications there have been several attempts to replicate and/or extend his work by other researchers. Due to the technical challenges of implementing a physical simulator, coupled with the lack of access to computing resources capable of comparable parallelization to the CM-5, it took several years before comparable results were achieved.

Taylor and Massey

The first published account of successfully reproducing Sims' work on standard PCs did not come until the work of Taylor and Massey (2000). These researchers re-implemented Sims' work by evolving morphologies and controllers for virtual swimming and locomoting creatures. By this time, off-the-shelf physics simulators that could run on standard PCs were available, which which removed a major obstacle to implementing such an evolutionary system. In particular, Taylor and Massey used the MathEngine physics engine which was freely available for academic use.

In addition to using an existing physics engine and running on standard PCs, Taylor and Massey's re-implementation differed from Sims' original in a few ways. The segments in their work were capped cylinders as opposed to cuboids which led to more efficient collision detection. Additionally, their joints were all ball and socket joints, and instead of applying forces directly between body parts as Sims did they used PD actuators in which the input to the actuator was interpreted as a desired orientation to the joint. Taylor and

Massey claimed that using PD controllers led to more rapid evolution of useful movements, although they did not present data to support this claim.

Besides these areas of difference, Taylor and Massey replicated Sims' work and were able to achieve interesting behaviors. Their work is of additional relavence for anyone interested in evolving robots in a simulated physical environment because they devoted a substantial amount of text to discussing issues related to handling numerical instabilities in the physics engine as well as discussing how fitness functions may be formulated to evolve robots that perform the desired task while avoiding undesirable behaviors that may result from seemingly intuitive fitness functions. Finally, they presented an overview of other work that had been conducted up to that point in time.

O'Kelly and Hsiao

O'Kelly and Hsiao (2004) presented another re-implementation of Sims' work. Their evolutionary system was much like Sims' competitive evolution, however instead of competing for possession of a box present in the environment as Sims' creatures did (Sims 1994a), their virtual robots "fought" in a virtual environment.[10] Specifically, the goal of each creature was to touch its opponent's root node before its own root node was touched. Their techniques were very similar to Sims', but in this case spheres were used instead of cuboids, and these spheres were connected by motors that spin along the axis of attachment similar to a wheel and axle, and were velocity controlled (neuron signals were interpreted as intended velocities where a positive value equaled counterclockwise motion and a negative value equated to clockwise motion). Additionally, they endowed every sphere with a number of different sensors: a binary touch sensor, sensors providing position and velocity of the sphere's motor, and sensors providing spherical coordinates to the enemy's root sphere as well as the enemy's nearest sphere.

Marbach and Ijspeert

Around the same time, Marbach and Ijspeert (2004) presented their *Adam* modular robot and simulation tool. This project was motivated by following the three major axes that "underlie the emergence of autonomous and self-organizing organisms in nature": phylogeny (evolution), ontogeny (development), and epigenesis (learn-

[10] In this work the open-source Open Dynamics Engine (ODE) (`http://www.ode.org`) was used for physical simulations.

ing). In this particular piece of work they concentrated on the phylogenetic axis–evolving the configuration of a homogeneuos modular robot along with its controller in a physical simulator[11].

This work is distinct from the others discussed in this section as the authors' goal was not to explore the entire space of possible morphological solutions, but rather to simulate robots composed of only one predefined module type based on those used in existing modular robotic systems. By using only a single module type the authors posit that their results should be easily transferable to physical robots, which in turn will benefit from the advantages of modularity, including versatility, adaptivity[12], and reliability (due to their redundancy).

While the author's goals were different, their methods were similar to others discussed in the section. A directed tree was used as the genotype, which was very close to Sims' directed graphs, except loops were not allowed (so it was not possible to reuse components recursively). Moreover, only hinge joints were allowed as the connections between modules. Instead of any form of neural controller, these joints were controlled by PD controllers driven by signals generated by harmonic oscillators with evolved amplitudes, frequencies, and phase shifts. Additionally, no sensory system was included so the control was strictly open loop.

Similar methods, including the same encoding, were used by von Haller et al. (2005) to evolve the morphology and control of underwater modular robots. In this work the controllers were neural central pattern generators (CPGs) that were first evolved in isolation and then instantiated in the robots. While the ability for evolution to control morphology was included, the authors state that simple and effective solutions were quickly found by the GA and adding or removing modules was hardly done. So in this case, allowing for the evolution of morphology did not allow for the improved performance or greater exploration of the search space as seen elsewhere, but neither did it prevent effective solutions from being discovered.

Chaumont et al.

Chaumont et al. (2007) also implemented a system based heavily on Sims' work (using ODE as the physics engine). Their publication presented another reproduction of Sims' work with minor modifications. Specifically the function set that they allowed for neuron activation functions was much smaller than that allowed by Sims, though time-dependent oscillators were still included. Additionally, instead of capping the number of allowed components in any one creature, the number of genes was capped instead (which, due to recur-

[11] In this work, once again, ODE was the physics engine employed.
[12] Modular robots should be versatile as they can be configured in many different ways and may adapt through self-reconfiguration.

sion, could still produce creatures of an uncapped number of components). Finally no garbage collection was used so that unconnected neurons could remain for future connections to be added. This feature seems more biologically plausible than the cleanup technique used by Sims, and it allows new connections to take advantage of previously disconnected parts without needing to recreate a whole new network part from scratch (although it does have the drawback of potentially creating large disconnected graphs).

Creatures were first evolved for locomotion. Here, as was the case with Taylor and Massey (2000), fitness evaluations were tailored to prevent simulation exploits, and a sizable amount of text was devoted to explaining the design decisions made as well as how to detect and throw out solutions which cause numerical instabilities. Additionally, the authors evolved creatures for two block-throwing tasks. In the first, just the distance that the block moved was selected for. In the second, accuracy at throwing towards a distant target was selected for. Creatures accomplished block-throwing through a given mechanism that was activated after a fixed amount of time or after receiving a neural signal. The block was given to the creatures *a priori* and was always attached to the root node; customized genetic operators were implemented that would not destroy the block-throwing mechanism.

Block throwing as a task was the most original contribution from this work. Several interesting block throwing creatures (or catapults, as the authors refer to them) were evolved with a variety of different strategies. Two of these strategies were the *drop-kicker*, which released its projectile, letting it fall to the ground and then kicking the projectile at the moment it touched the ground, and the *acrobat*, which began standing on the projectile and then performed a somersault, releasing the projectile at the correct time. Both of these examples demonstrated synchronization between the controller and an event in the environment. Videos of these strategies and others can be seen at `http://public.kgi.edu/~nchaumon`.

Lassabe, Luga and Duthen

Lassabe, Luga, and Duthen (2007) presented another extension of Sims' work (see also Luga (2008)). They employed graph grammars (graphtals) similar to Sims' in order to evolve simulated creatures composed of cuboids. However, instead of using augmented neural network controllers, their creatures were controlled by classifier systems. These work by initially generating a database of 1000 randomly-generated patterns and then evolving classifiers which take sensory input and then output a set of rules corresponding to patterns in the database. These patterns were composed and used by the effectors controlling the creatures' muscles.

The other significant deviation from Sims' work is that Lassabe et al. devote more focus to the environments[13] in which the creatures evolved, which allowed them to generate more complex behaviors. Specifically, while walking on a flat floor was evolved first (similar to other works) they went on to evolve walking in a specific direction, walking across trenches, climbing stairs, walking across terrain with hills and valleys, skateboarding[14], and finally cooperation–where two clones of the same creature were put in the same environment and tasked with pushing a block that was too large for a single individual to push on its own. Additionally, the authors discussed their plans for extending these complex environments into a full-fledged virtual ecosystem with plants and various forms of creatures. More details of extending their work in this direction were presented in (Lassabe et al. 2006).

The more complex environments under which evolution was carried out in this work indicate promising directions for the evolution of morphology and control of virtual robots. Through environmental complexification, novel behaviors were achievable on tasks not previously reported in the literature. A video of the best creatures evolved in this work is available from these authors at `http://www.irit.fr/~Nicolas.Lassabe/videos/alife2007.mov`.

Miconi and Channon

Another replication of Sims' work was carried out by Miconi and Channon (2005)[15]. The main novel contributions of their work was the utilization of standard neurons (using sigmoid or hyperbolic tangent activation functions) in contrast with Sims' arbitrary function set. Additionally, loops were not allowed in the genome, so no explicit segmentation was allowed (but bilateral symmetry could still be enforced through the use of a reflection flag). Furthermore, instead of specifying a force or torque, the actuators which they utilized would specify a desired speed–thus modeling servomotors. Moreover, this work employed a steady state GA as opposed to the generational GA usually used.

Of greater interest is a subsequent article by Miconi (2008b) where once again Sims' graph encoding, restricted to using standard activation functions, was used. Here loops/segmentation were added back in as well as a mechanism to allow for fine grained neural wiring among replicated limbs for asymmetric informa-

[13] In this work the Breve ALife simulator (Klein 2003), also based on ODE, is used for the simulations.

[14] The environment was equipped with a form of skateboard placed next to the creature, and selection was for distance moved with the skateboard.

[15] Though it is not stated explicitly, from the images included in their paper it appears that in this work ODE is once again employed as the physics engine.

tion flow. Like (O'Kelly and Hsiao 2004), a co-evolutionary setup was used to evolve creatures for fighting, but instead of merely having a designated target on each individual a more realistic approach of modeling physical damage was used. Fitness was then a ratio of $\frac{\text{damage inflicted on opponent} - \text{damage sustained}}{\text{total damage inflicted on both creatures}}$. Damage was determined through the use of freezing the simulation and progressing creatures individually through the simulator one step at a time to determine how their movements would affect penetration depth. The more that one creature's movements caused it to penetrate the other the greater the inflicted damage on the other. This methodology of simulating physical conflict is quite novel, and may be useful in more realistically replicating behaviors of biological organisms. Additionally, these researchers have made available all of their code under an open-source GPL license, which may be a useful resource for future research in this area.

Krčah

Krčah (2008) presented his Evolving Robotic Organisms (ERO) system, which also used Sims' graph encoding to evolve creatures for phototaxis, swimming, jumping, and walking[16]. As opposed to the GAs employed by previous researchers, he utilized an extension of the NeuroEvolution of Augmenting Topologies (NEAT) method introduced by Stanley and Miikkulainen (2002) (NEAT will be discussed in greater detail later in this dissertation). The extension, Hierarchical NEAT (hNEAT), maintained hierarchical historical markings both on nodes of the top-level graphs describing morphology and on the sub-graphs describing the neural circuitry. In this way, crossover could use the historical markings on the morphological nodes, and when corresponding body parts were found, crossover could additionally operate on the neural networks based on their own historical markings. This allowed for recombination on the level of individual neurons and neural connections, which was not possible with previous approaches. Additionally, it allowed species to form not only according to the differences in their morphology, but also according to the differences in their neural networks. This algorithm was shown to reach predefined fitness thresholds significantly faster than a standard GA on all four tasks investigated, and thus presents a promising direction for future work in this area.

Lehman and Stanley

More recently, Lehman and Stanley (2011b) employed Krčah's ERO system to evolve a diversity of locomoting creatures within a single run instead of converging to a single optimum. To do this they employed a multi-

[16]Once again the ODE physics simulation engine was utilized.

objective evolutionary algorithm (Fonseca and Fleming 1993, Deb 2001) to combine novelty search (Lehman and Stanley 2011a) with a local competition objective. They measured the height, mass, and number of active joints of each morphology in order to define a three-dimensional Euclidean space on which their novelty metric could be calculated. They then selected for diversity in this space along with a local or global fitness based on the creatures' abilities to locomote. While using global competition was better able to find the maximum fitness, local competition was better able to exploit a wide range of morphological niches by evolving successfully locomoting creatures in each niche. Techniques such as these demonstrate not only how it is possible to evolve a diversity of successful creatures within a single run, but also may be useful for escaping local optima in order to ultimately reach higher fitness peaks. This idea has been deeply explored in the evolution or robot controllers by Mouret and Doncieux (2012).

Direct Encodings

While Sims' graph-based encoding (or variants there-of) have been very popular techniques as evidenced by the number of research projects in this area, it is by no means the only possibility for jointly evolving the topology of robots with their control policies. More complex, biologically motivated developmental models exist which will be discussed in the next subsection, but first several methods which take a simpler, direct approach are presented.

Lipson and Pollack

Lipson and Pollack (2000) presented a system where robots with morphologies and neural controllers represented by vectors were evolved in simulation to locomote. Sufficiently fit robots were then semi-automatically fabricated with rapid prototyping (3-D printing) technology. Their robots were composed of variable-length bars connected at vertices by ball and socket joints. The genome was of variable length so creatures containing different numbers of components could be represented. Robots were evolved in a quasi-static kinematic simulator for locomotion, which reduced the possible forms of locomotion that could be evolved (they thus disallowed evolution of dynamic gaits). Additionally no sensors were used, so behavior was limited to open-loop control. However, this work was quite novel because it represented the first instance of automatically manufacturing evolved morphologies. In this case, the only step needed to be taken by humans was attaching stepper motors and a micro controller running the neural network after the robot came out of the printer.

Bongard and Paul

Around this same time Bongard and Paul (2000) employed a direct genotype to phenotype mapping to evolve simulated robots. The genotypes were variable length bit strings. The robot phenotypes were composed of spheres connected with links along with embedded sensor, motor and internal neurons. The authors selected for directed locomotion in conjunction with either bilateral symmetry or bilateral asymmetry. Using the results of these two sets of runs they were able to demonstrate how bilaterally symmetric creatures are more efficient based on multiple measures of efficiency. This result gave further justification for why symmetry may be useful beyond the fact of its observed prevalence in biological organisms. Therefore, it may be useful to include mechanisms which introduce an explicit bias towards bilateral symmetry when evolving robot morphologies.

Macinnes

Macinnes (2003) evolved robot morphologies and continuous time recurrent neural network (CTRNN) controllers (Beer 1996) for a visually guided directed locomotion task[17]. Once again a direct genome representation was used, divided into two sections: one for morphology and one for the neural network. The robots were composed of variably sized blocks, and their goal was to locomote towards a blue sphere by means of a vision sensor with a genetically defined angle of projection. The authors claimed that it was not possible to evolve this directly, and so made use of incremental evolution which first evolved for general locomotion, then for approaching a wall based on the output of a directional beacon, then gradually (over evolutionary time) swapping in the vision sensor for the directional beacon, then reducing the size of the wall until it became a cube, and finally for approaching the colored cube when it was placed in arbitrary locations. The following year, Macinnes and Paolo (2004) extended this work by limiting search to buildable morphologies using components from a pre-defined LEGO parts library. These robots were evolved for locomotion, and one evolved robot was physically instantiated to demonstrate how topological evolution could be used to create LEGO robots of arbitrary configurations.

[17] Robots simulated in ODE.

Shim and Kim

Also in 2003, Shim and Kim took on the novel task of evolving flying creatures for directed flight and hover ability. Their morphologies consisted of a fixed component (central fuselage) and a topologically evolvable component. Symmetry was enforced by duplicating the evolvable component on either side of the fuselage. The genotypes they employed were lists which defined the connectivity of structural elements (modeled as capped cylinders). Triangular rigid films were attached between connecting components to allow for flight. They used augmented neural networks as controllers with various functions, but without weighted connections or biases. The authors were able to evolve several different classes of flying structures, including those with short wing spans which flapped rapidly, and those with longer wing spans and slower flapping.

Hara and Kikuchi (2003)

Hara and Kikuchi (2003) also presented related work. They used genetic programming to evolve the morphology and control of simulated robots for a garbage collection task. The morphologies were composed of omni-directional wheels, each surrounded by a variable sized ring. The rings were connected together with rods. A direct graph representation of this morphology was used, and controllers were evolved Lisp programs. The researchers were able to achieve a variety of results. However, it should be noted that this as well as all other wheeled robot projects have severe limitations in that they are limited to acting on relatively even terrain.

Komosinski and Rotaru-Varga

As mentioned earlier, in addition to the static structure height maximization task, Komosinski and Rotaru-Varga (2002) also investigated using their Framsticks system for evolving agents on dynamic tasks. These tasks included active height maximization and locomotion velocity. For both of these tasks agents were equipped with neural network controllers. The neural networks consisted of standard units with sigmoidal activation functions plus an extension: each neuron possessed an internal state which updated with a certain inertia. The neural networks received inputs from a variety of sensors including orientation and touch, and could actuate the robots through two type of effectors (muscles): bending and rotating.

While the active height maximization fitness criterion occasionally resulted in creatures which purposefully moved, the results were similar to the static case, because the static structures were close to optimal.

When evolved to maximize locomotion velocity the more abstract encodings did tend to produce more biologically realistic designs. However, they constrained their experiments to operate on a small number of sticks, and so there were not qualitative differences in the performances between their direct and indirect encodings. This suggests that perhaps the allowed complexity of evolved agents must be above a certain threshold before the greater evolvability of more abstract encodings becomes evident.

Generative Encodings

While there may be advantages to using simple, direct encodings like those just discussed, they are not without their drawbacks. Of particular importance is scaling. With a direct encoding where a "gene" is needed for every component, the size of the genome necessarily grows linearly (or worse) with the number of components. Biological organisms, on the other hand, represent an immense amount of complexity within relatively small DNA encodings. Inspired by this, many researches have used generative grammars for evolving robots which can more compactly represent complex morphologies and controllers. These methods are related to Sims' graphs with recursion, but involve specific rewrite rules and other techniques and they are discussed here separately.

Hornby and Pollack

Hornby and Pollack (2001b) (see also Hornby and Pollack (2001c) and Pollack et al. (2001)) presented the first use of L-systems for evolving the morphology and control of simulated 3-D robots. This work was an extension of their previous work evolving physical structures (Hornby and Pollack 2001a) which was discussed Sect. 1.3. Here however, instead of defining structures as voxels through which the turtle travels, each movement of the turtle through unexplored space resulted in the placement of a bar. Special commands were included which allowed for the construction of one of many types of joint at the terminal of this bar. They first explored using oscillating motors to control the evolved robots by evolving parameters of the joint creation function. These parameters specified the joints' rate of oscillation and included special commands to modify the phase offset of a given oscillator.

After experimenting with these oscillators, Hornby and Pollack further extended their system to incorporate neural networks. They did this by combining their command language for creating morphology with one that operated on a neural network. Output neurons were automatically created any time a joint was added,

and the stack of build states was augmented to include the current link in the neural network which was being operated on. They allowed for standard sigmoid and hyperbolic tangent activation functions as well as neurons with linear activation functions and stateful oscillators (which biased the network toward cyclic behavior). Simulated robots were evolved for locomotion in the quasi-static kinematic simulator developed by Lipson and Pollack (2000). Again, while this work was quite novel it did not allow for dynamic motion or closed-loop control (where control signals are modulated through sensorial feedback). However, the authors were able to demonstrate a significant increase in fitness and an order of magnitude greater number of components of robots evolved through this method compared to a direct encoding.

Later, these authors, together with Hod Lipson, extended this work further by combining robots evolved from the generative L-systems with physical fabrication (Hornby et al. 2003). They were able to fabricate evolved robots made of bars and joints and actuated by oscillators. This was done both through the use of a predefined set of components (bars of regular length and fixed or actuated joints) and by means of rapid prototyping technology.

1.4.3 Developmental Models

Even though grammars may be able to compactly represent an elaborate structure, they are an abstraction that only approximates aspects of biological organisms: they do not model how organisms develop in nature. Inspired by biology, some researchers have gone further to create more biologically plausible models of growth based on cell division, differential gene expression and genetic regulatory systems.

Dellaert et al.

Early work in this area was performed by Dellaert et al. (1996), who presented two developmental models for evolving 2-D simulated creatures. Both models were based on cell division and differentiation driven by a form of genetic regulatory network (GRN). While the topology of each agent was evolved, it was restricted to subdivisions of an initial square “egg” cell in both cases. With their first, more biologically plausible model they were unable to evolve complex behaviors from scratch, but demonstrated it through the use of a hand designed genome. Their second model based on random boolean networks (RBNs) was able to evolve line following behavior from scratch.

Similarly, Mautner and Belew evolved simulated 2-D robots controlled by neural networks through cell division (Mautner and Belew 2000). Their approach however was based on grammars (similar to those discussed previously) which defined cell replacement rules and so can be considered a hybrid approach. Since this work was confined to 2-D simulations it will not be discussed in detail here.

Eggenberger

Eggenberger's work evolving shapes based on differential gene expression discussed above also fits into this class of system. He subsequently extended this to allow for growing neural networks (Eggenberger 2003). With this system he was able to demonstrate how a single neuron could grow axons to a specific target region, and how a pre-existing (two-dimensional) robot could have its shape optimized while maintaining its behavior. Once again, since this work was limited to two dimensions it will not be discussed in detail[18].

Bongard

Of greater interest is Bongard's work where he used a developmental encoding (Artificial Ontogeny) to evolve morphologies and controllers of simulated 3-D robots (Bongard 2002a, Bongard and Pfeifer 2003)[19]. This work could be considered an extension of Eggenberger's work as it employed similar models of GRNs. In this work, robots composed of cylindrical (Bongard 2002a) or spherical (Bongard and Pfeifer 2003) structural units connected by active or passive single degree of freedom joints were grown from a single initial structural unit and a simple neural network. Based on the concentration of gene products these units were able to grow, add or subtract embedded neurons and synapses, move neurons, and split into multiple units. Specifically, units would grow until twice their initial size, split, and would then connect to a new unit at one of six diffusion sites.

Evolved genomes encoded the properties of a GRN that dictated how different gene products affected the expression of other genes. Specifically, every time a new structural unit was created it was equipped with a touch sensor neuron, a motor neuron, and a synapse connecting the two. Certain gene products affected the embedded neural network in a variety of ways. Of particular note is that they could cause neurons to

[18]There have been a number of other studies employing various generative and developmental models for evolving 2-D creatures. These include recent work by Schramm et al. (2011) and Joachimczak and Wróbel (2012), but since this review is focused on evolving 3-D morphologies they will not be discussed further here.

[19]The MathEngine physics simulator was used for this work.

migrate. When this happened their synapses would travel with them, thus allowing for distally connected neural structures.

Creatures were grown using these techniques and evaluated in a simulated environment. In (Bongard and Pfeifer 2003) two tasks were investigated: directed locomotion and block pushing. In (Bongard 2002a) the task was locomotion. It was shown that more successful creatures evolved to have more modular GRNs: the genes controlling morphology and the genes controlling neurology evolved to not have large effects on each other. It was additionally demonstrated how in block pushing robots functional specialization had evolved. In the central part of the robots many structural units lost their motor capability during growth as a result of the migration of motor neurons out of these units, while sensor units did not migrate out of these units. This suggested that these sensors may have directed distal motors. Furthermore, agents with repeated, differentiated gene expression patterns were found, which demonstrated that it was not necessary to have an explicitly recursive encoding to achieve such structures.

Open Ended Evolutionary Worlds

As opposed to evaluating agents on an objective function either in isolation or in competition with a single other agent it is also possible to have an unconstrained simulated 3-D environment composed of artificial creatures. In these environments a fitness measure may still be used to guide reproduction as described by Komosinski (2000), or reproduction and death may be directly modeled based on simulation interactions. This was the approach taken by Spector et al. (2007). In their work, creatures were composed of "division blocks." These could shrink, grow, reproduce, live and die based on energy and waste (either gleaned from a virtual sun or acquired from other division blocks). These blocks each had a number of predefined sensor and effectors and were equipped with recurrent neural networks. This was similar to the virtual ecosystem idea of Lassabe et al. (2006) mentioned earlier, but being an open ended evolutionary system no specific behavior was selected for, so it is questionable how much it relates to designing useful robots. (131)

1.5 Evolution of Complexity

The evolution of complexity is a topic far too broad to do justice to within the confines of this chapter. That being said, it is a subject to which this dissertation hopes to contribute, and so some relevant literature will be

briefly touched upon in this section. Particular focus will be placed on how evolutionary robotics and digital evolution may contribute to this discussion.

Whether and when increased complexity evolves is a longstanding open question in biology. Many researches take it as a given that complexity has increased over evolutionary time. This notion is summed up by Bedau (1998), who states

> *The progression of evolution in our biosphere seems to show a remarkable overall increase in complexity, from simple prokaryotic one-celled life to eukaryotic cellular life forms with a nucleus and numerous other cytoplasmic structures, then to life forms composed of a multiplicity of cells, then to large-bodied vertebrate creatures with sophisticated sensory processing capacities, and ultimately to highly intelligent creatures that use language and develop sophisticated technology. (p. 131)*

This is an idea with deep roots, going back at least as far as Lamarck (1809). However, others are less inclined to take this point of view. For example, McShea (1996) begins by stating "The notion that complexity increases in evolution is widely accepted, but the best-known evidence is highly impressionistic." He goes on to conclude "Something may be increasing. But is it complexity?"

Much of this confusion stems from the way in which complexity is defined. Complexity is a very broad term that different researchers have interpreted in different ways. These include the number of different cell types (Bonner 1988), depth of hierarchical organization (Maynard Smith 1988), description length (Kolmogorov 1965), and information content (Shannon 1948), among others. For a good overview of the different meanings of complexity in relation to evolution see (McShea 1991, McShea 1996, Feldman and Crutchfield 1998, Adami 2002, Miconi 2008a).

Additional confusion comes from the ambiguity of what it means for there to be a tendency or trend. As discussed by McShea (1994) (also (McShea 1996, McShea 2005)) and further elaborated upon by Miconi (2008a) trends can be passive or driven. Passive trends may result from envelope expansion without any directional bias. If there is a minimum level of complexity necessary for life but no upper bound, then simply through unbiased random walks complexity will increase over time. A trend will then be evident in both maximum and mean complexity. On the other hand, driven trends exhibit a consistent, directional bias.

Notwithstanding this ongoing debate, and the many meanings of complexity, it is quite clear that evolution has produced a diversity of organisms possessing the types of intelligence that would be desirable to

embed in robots. This begs the question: under what circumstances will complexity increase over evolutionary time? Fortunately, robotics and artificial life provide experimental test-beds for investigating various complexity relationships that would be difficult or impossible to perform with biological organisms. Two specific examples will be described here.

Lenski et al.

Lenski et al. (2003) employed the Avida digital evolution system to investigate the evolution of complex features. Individuals in Avida are computer programs (circular sequences of instructions), which compete for the energy required to execute their instructions. In their experiments, individuals were given energy in proportion to their genome (sequence) length, and were able to gain further energy by executing specific logic functions (with rewards scaled exponentially with function complexity). Starting with a population of individuals capable of replication but without the ability to execute any logic functions, the organisms evolved through competition for energy which allowed them to execute their instructions. These individuals were able to evolve the ability to compute the most complex function (*Eq*) when rewarded for simpler functions, but this ability was independent of being rewarded for any specific simpler function. However, when there was no reward for any simpler function, *Eq* was never evolved. This work demonstrated that complex features generally evolve by modifying existing structures, and that which complex features will evolve is dictated by the organisms' environment (the functions for which they were rewarded, in this case). Moreover, by using digital organisms and therefore having access to the complete genetic history of evolved individuals they were able to analyze the specific mutations which led to the emergence of complex features.

Paul

Paul (2006) used a GA to evolve gaits for a 24-degree of freedom tensegrity robot, by only using two, three, or four control inputs. Due to the high complexity of the morphology–particularly the large amount of dynamic coupling present in the tensegrity design–these few control inputs were able to give rise to complex and non-linear dynamics capable of producing successful gaits. The author described this process as a form of *morphological computation* where the complexity of the controller is offloaded to the physical structure of the morphology. It also serves as an instance of what Pfeifer and Scheier (1999) referred to as the *morphology and control trade-off*. While this paper was not directly concerned with the evolution of complexity, it did

demonstrate how the complexity of an evolved controller or neural system is influenced by the complexity of the morphology that it controlls. These ideas will be used as inspiration in this dissertation for investigating how the complexity of evolved morphologies vary in relation to the environments within which they evolve.

1.6 Conclusions

This chapter has examined a diverse set of publications detailing work where aspects of a robot's morphology were evolved (often in addition to its control architecture). These works span a broad spectrum of techniques, motivations and scope. While there are most likely a few publications that have been overlooked, this chapter has attempted to cover the majority of research done in this area.

Most explorations were conducted entirely in simulation, which has the advantage of being able to rapidly construct and evaluate different solutions, possibly in parallel. With the ongoing increase in computing power per dollar it should be possible to continue increasing the scope of what is possible to simulate in the future– including more diverse environments with interacting agents. However, simulation does have its drawbacks. One of the main criticisms of the work of Sims and his intellectual descendants is that morphologies may be simulated that are impossible to physically construct. Moreover, even if the simulation is designed to only allow for plausible morphologies, it is often difficult to transfer the results to a physical robot. That being said, the works of Lipson, Pollack, Hornby, Lund and their ilk have demonstrated that it is possible to fabricate morphologies evolved in simulation either through rapid prototyping or with predefined parts if care is taken.

The genome representation utilized is also of great import when developing such systems. Much work has demonstrated how generative or developmental models yield more evolvable systems than those using comparable direct encodings. Additionally, such indirect encodings can more compactly represent a complex robot morphology and controller, thus allowing for the evolution of vastly more complex robots. It is likely that if these methods are going to scale up to producing robots which will perform meaningful tasks in everyday environments that such encodings will be necessary.

Moreover, it is interesting to consider the task environments in which agents have been evolved. Land-based locomotion has been the most common. Obstacle avoidance or similar tasks such as garbage collection with wheeled robots were also common. Less common were swimming or aerial motion, though with proper

simulation underwater movement can be easily evolved. Evolution of competing or fighting creatures were also investigated by several researchers, and more novel tasks such as block throwing and block pushing have been investigated. Of particular note are the more complex environments and tasks investigated by Lassabe et al. (2007). Their inclusion of environmental features such as blocks to walk on, or skateboards to move with were quite novel and represent a promising direction for realizing more complex morphologies and behaviors.

Finally, Section 1.5 briefly discussed some questions surrounding the evolution of complexity, and gave some examples of how artificial life and evolutionary robotics have made contributions to this domain.

1.7 Dissertation Outline

The remainder of this dissertation draws from all of these previous contributions to investigate the challenges surrounding automatically producing complex robot morphologies and controllers by means of evolutionary algorithms.

One such challenge is how to optimize a controller that can orchestrate dynamic motion of different parts of the body for different behaviors. Chapter 2 describes how an incremental shaping method (Dorigo and Colombetti 1994, Saksida et al. 1997, Bongard 2008b) (also know as scaffolding) can be used to address this challenge. Specifically, robot controllers are evolved to both coordinate a robot's leg motions to achieve directed locomotion toward an object, as well as control a gripper to manipulate the object once it has been reached. It is shown that success is dependent on the order in which these behaviors are learned, and that despite the fact that one robot can master these behaviors better than another with a different morphology, this learning order is invariant across the two robot morphologies investigated.

In Chapter 3, the question as to how to distribute responsibility for multiple behaviors across an agents's controller and morphology is investigated. A robot is trained to locomote and manipulate an object, but the assumption of functional specialization is relaxed: the robot has a segmented body plan in which the front segment may participate in locomotion and object manipulation, or it may specialize to only participate in object manipulation. In this way, selection pressure dictates the presence and degree of functional specialization rather than enforcing such specialization *a priori*. It is shown that for the given task, evolution tends to produce functionally specialized controllers, even though successful generalized controllers can also

be evolved. Moreover, the robot's initial conditions and training order have little effect on the frequency of finding specialized controllers, while the inclusion of additional proprioceptive feedback increases this frequency.

Chapter 4 presents a novel method for evolving 3-D physical structures encoded with CPPNs for a dynamic task. This method is shown capable of producing physical artifacts that capture the non-obvious yet close relationship between function and physical structure. Moreover, it demonstrates how more fit solutions can be achieved with less computational effort by using feedback mechanisms within the growth procedure as well as incremental changes in object resolution over evolutionary time.

In Chapter 5 this method is extended to simultaneously evolve the morphologies and controllers of actuated robots using CPPN-NEAT (Stanley 2007). This new method is also capable of dynamically adjusting the resolution at which components of the robot are created: a large number of small components may be present in some body locations while fewer, larger components are present in other locations. Advantages of this capability are demonstrated on a simple task, and implications for using this methodology as a form of scaffolding to create more complex robots are discussed.

These methods are further extended in Chapter 6 in order to allow for CPPNs to be translated into complete robots including their physical topologies, sensor placements, and embedded, closed-loop, neural network control policies. It is shown that this method can evolve robots for the given task. Additionally, it is demonstrated how the performance of evolved robots can be significantly improved by allowing recurrent connections within the underlying CPPN genomes. The resulting robots are analyzed in the hopes of answering why these recurrent connections prove to be so beneficial in this domain.

A new method of constructing actuated robots from CPPNs is introduced in Chapter 7. This method allows for the construction of more complex morphologies built out of triangular meshes. This method is used to investigate the relationship between morphological complexity and the complexity of a given task environment. It is hypothesized that the morphological complexity of a robot should increase commensurately with the complexity of its task environment. This hypothesis is tested by evolving robot morphologies in a simple environment and in more complex environments. More complex robots (based on an information theoretic measure of complexity) tend to evolve in the more complex environments, lending support to this hypothesis.

These ideas are further explored in Chapter 8, where instead of the *morphological complexity* of robots with a fixed number of degrees of freedom, their *mechanical complexity* (the number and range of a robot's

mechanical degrees of freedom) is investigated in relation to their task environments. Counter to intuition it is found that mechanical complexity decreases in more complex task environments. This demonstrates that these different forms of complexity do not necessarily correlate with each other, but are likely orthogonal.

Chapter 9 takes the results from Chapter 7 and investigates their causes in greater detail. Specifically, the ways in which complexity varies over evolutionary time is compared across environments, as well as against neutral shadow models (Bedau et al. 1998, Rechtsteiner and Bedau 1999) which are free from environmental influences. It is shown that complex environments select for increased morphological complexity, while morphological complexity is constrained in simple environments. These results are corroborated by additional experiments employing a multi-objective selection mechanism to select for morphological simplicity in addition to behavioral competency. Under this regime, the differences between the morphological complexities of organisms evolved in simple versus complex environments become even more pronounced, further supporting environment's role in selecting for complexity.

The results of these previous studies suggest that gradually increasing the complexity of task environments may provide a principled approach to evolving more complex robots. But, how exactly these environments should vary over time is not clear. One idea is to use co-evolution in order to evolve environments and robots together in such a way that their respective complexities increase as appropriate. In Chapter 10, the final chapter, ongoing experiments investigating this possibility are discussed along with additional directions for future research.

Chapter 2

How Robot Morphology and Training Order Affect the Learning of Multiple Behaviors

Automatically synthesizing behaviors for robots with articulated bodies poses a number of challenges beyond those encountered when generating behaviors for simpler agents. One such challenge is how to optimize a controller that can orchestrate dynamic motion of different parts of the body at different times. This paper presents an incremental shaping method that addresses this challenge: it trains a controller to both coordinate a robot's leg motions to achieve directed locomotion toward an object, and then coordinate gripper motion to achieve lifting once the object is reached. It is shown that success is dependent on the order in which these behaviors are learned, and that despite the fact that one robot can master these behaviors better than another with a different morphology, this learning order is invariant across the two robot morphologies investigated here. This suggests that aspects of the task environment, learning algorithm or the controller dictate learning order more than the choice of morphology.

2.1 Introduction

Robots with three-dimensional, articulated bodies that act in physical or physically-realistic environments must be able to coordinate motion of different subsets of their body parts during different phases of performing a task. In this work a behavior is defined as the successful coordination of one of these subsets to achieve part of a desired task. Ideally, the same controller should be able to direct these different behaviors and allow transitions between them.

Evolutionary robotics (Harvey et al. 1997, Nolfi and Floreano 2000) is an established technique for generating robot behaviors that are difficult to derive analytically from the robot's mechanics and task environment. In particular, such techniques are useful for realizing dynamic behaviors (eg. (Reil and Husbands 2002, Hornby et al. 2005)) in which individual motor commands combine in a nonlinear fashion to produce behavior, thereby making analytical derivations of optimal controllers infeasible. However, evolutionary algorithms alone are often not sufficient to evolve multiple dynamic behaviors: to date most reported efforts have primarily focused on realizing a single behavior, such as locomotion (Reil and Husbands 2002, Hornby et al. 2005) or grasping (Fernandez Jr. and Walker 1999, Chella et al. 2007).

Previous work has shown that it is possible to realize multiple behaviors in a robot by gradually incorporating more modules into its controller (Brooks 1986, Calabretta et al. 2000). However, this approach does not scale well as the number of modules, and therefore the size of the controller grows with the number of behaviors. A scalable approach to behavioral flexibility might allow the same dynamic controller to exhibit multiple attractor states, in which individual behaviors correspond to individual attractor states, an idea with some currency in the robotics literature (Inamura et al. 2004, Okada and Nakamura 2004). One of the main difficulties in this approach however is realizing multistability (Foss et al. 1997) in the controller: it should settle into different attractor states that correspond to the different desired behaviors in the face of the appropriate sensory stimulation. Another recent finding indicates that rather than different behaviors corresponding to different attractor states, they may correspond to distinct transients within the dynamical system composed of the agent's environment, body and brain (Izquierdo and Buhrmann 2008).

This paper extends the results reported in (Bongard 2008) in which a virtual legged robot was trained to perform a mobile manipulation (Carriker et al. 1991, Seraji 1998) task. The robot in (Bongard 2008) learned to coordinate its legs to locomote toward an object and then coordinate the motions of a gripper to

achieve object manipulation. It was demonstrated there that successful attainment of both these behaviors is dependent on the order in which they are learned. This result lends support to the growing body of evidence that incremental shaping ((Singh 1992), (Dorigo and Colombetti 1994) and (Saksida et al. 1997)) – the gradual complexification of an agent's task environment, also known in the developmental psychology literature as scaffolding (Wood et al. 1976) – can improve the probability of successful learning. However, the selection of an appropriate scaffolding schedule that enforces the order in which behaviors should be learned greatly impacts the probability of the agents successfully learning all of the behaviors (Beer 2008). The question then arises as to what dictates this learning order: the task environment, the learning algorithm, the controller, the robot's morphology, or some combination of all four.

In the work presented here the dynamic scaffolding method described in (Bongard 2008) is extended to enable a virtual autonomous robot to overcome three learning milestones: object manipulation, dynamic forward legged locomotion toward an object, and directed legged locomotion toward an object, all using a single monolithic controller – a feat, insofar as the authors are aware, that has not been previously reported in the literature. It is shown that, from among several scaffolding schedules that attempt to train the robot to achieve these behaviors in different orders, that the one that selects for manipulation, then forward locomotion, and then directed locomotion significantly increases the probability of a robot successfully learning all three, and that this order is invariant across two different robot morphologies that were investigated. In the next section the virtual robots and the incremental shaping method are introduced; the following section reports results demonstrating how this method, with the proper scaffolding schedule, can produce controllers that succeed in previously unseen environments, and the final sections provide some discussion and directions for future investigation.

2.2 Methods

This section first describes the two virtual robots used for this work followed by a description of their controllers. Next the incremental shaping algorithm used for training the robots is presented along with the various dynamic scaffolding schedules investigated here. The section concludes with a description of the metrics used to evaluate the robots' success.

2.2.1 The robots

In this work two virtual quadrupedal robots are used[1]. **Robot 1** (Fig. 2.1, left) is comprised of a main body, four legs and a front gripper. Each leg consists of an upper and lower part connected to each other and the main body. The gripper is comprised of a small spherical base connecting the main body to the gripper pincers. The gripper base can be rotated upward relative to the main body, and both the left and right pincers are comprised of a gripper arm (proximal to the gripper base) and gripper tip (distal to the gripper base). This robot is identical to the one used in (Bongard 2008) and the reader is referred there for more details regarding the robot's morphology.

Robot 2 (Fig. 2.1, right) is identical to robot 1 except for the orientation of the legs. Robot 2 has been modified by rotating the legs at the point they are attached to the main body such that each is positioned at a 45° angle to the main body. The upper legs in this robot move vertically in the plane defined by the vector lying along the upper leg and a downward-pointing vector, while the lower legs continue to move in the sagittal plane. This alteration was implemented to make turning easier.

Eight motors actuate the four upper and lower legs, another motor actuates the gripper base, and four motors actuate the base and distal parts of the left and right gripper pincers, for a total of 13 motors. A touch sensor and distance sensor reside in both the left and right gripper tips, a rotation sensor resides in the gripper base, and a distance sensor resides on the robot's back, for a total of six sensors. The touch sensors return a value of one when the corresponding body part touches another object and zero otherwise. The distance sensors return a value commensurate with the sensor's distance from the target object: they return zero if they are greater than five meters from the target object and a value near one when touching the target object. Object occlusion is not simulated here; the target object can be considered to be emitting a sound, and the distance sensors respond commensurately to volume.

The robots attempt to locomote toward, grasp and lift a rectangular target object that is placed at varying locations in relation to the robot. Unlike the robot's task in (Bongard 2008), in this work the target object is not constrained to being placed in front of the robot within its sagittal plane: additional target object placements away from the robot's centerline select for turning behavior.

[1]These results have not yet been validated on a physical robot, as the multiple morphologies would require constructing a morphologically-reconfigurable legged robot. However, this option will be explored in future work.

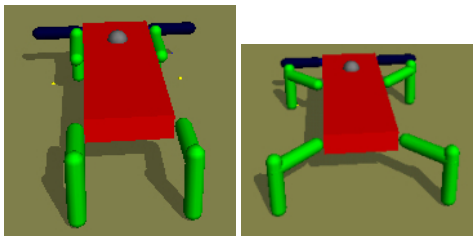


Figure 2.1: The two virtual robots used in this work. Left: **Robot 1**, right: **Robot 2**.

2.2.2 The controllers

Each robot is controlled by a continuous time recurrent neural network (Beer 2006). The CTRNN is composed of 11 motor neurons (the two gripper arm motors share the same motor neuron, as do the two gripper tip motors to ensure the gripper closes symmetrically). The remaining 9 motors each receive commands from their own motor neuron. Other network configurations such as those containing non-motor or hidden neurons were experimented with, but are omitted from the current work, because they were not found to improve performance.

The value of each motor neuron is updated according to

$$\dot{y_i} = \frac{1}{\tau_i}\left(-y_i + \sum_{j=1}^{11} w_{ji}\sigma(y_j + \theta_i) + \sum_{j=1}^{6} n_{ji}s_j\right) \quad \text{for } 1 \leq i \leq 11 \tag{2.1}$$

where y_i is the state of neuron i, w_{ji} is the weight of the connection from neuron j to neuron i, τ_i is the time constant of neuron i, θ_i is the bias of neuron i, n_{ji} is the weight of the connection from sensor j to neuron i, s_j is the value of sensor j and $\sigma(x) = 1/(1 + e^{-x})$ is the logistic activation function.

The virtual robot with a given CTRNN controller is evaluated over a set number of simulation steps in a physical simulator[2]. For each simulation step, using a step size of 0.0005, the sensors, CTRNN, joint torques and resulting motion are updated.

[2]Open Dynamics Engine: www.opende.org

2.2.3 Training

A version of incremental shaping extended from the algorithm presented in (Bongard 2008) is used for dynamically tuning the robot's task environment to facilitate learning. This method is outlined in Fig. 2.2. A random CTRNN is created by choosing all τ from the range [0.1, 0.5], all w from [-16, 16], all θ from [-1, 1], and all n from [-16, 16]; these ranges were found useful in previous work (Bongard 2008). This gives a total of $11 + 11 * 11 + 11 + 6 * 11 = 209$ evolvable parameters. The robot is then equipped with this controller and allowed to behave in a task environment for 100 time steps in which the target object is placed directly in front of the robot. After evaluation the fitness of the controller is computed as

$$f_{sub} = \max_{k=1}^{t}(D(LeftgripperTip, k) * D(RightgripperTip, k)) \tag{2.2}$$

if the touch sensors in the left and right gripper tips fail to fire at the same time during any time step of the evaluation period, and

$$f_{sub} = 1 + \max_{k=1}^{t}(D(SensorNode, k)) \tag{2.3}$$

otherwise, where t is the evaluation time, and $D(x, k)$ indicates the value of the distance sensor affixed to body part x at time step k. Eqn. 2.2 rewards controllers for steering the robot toward the target object. Eqn. 2.3 rewards controllers for also lifting the target object onto the robot's back (where the sensor node is located) while it is touching the target object with both gripper tips.

One extension added to the algorithm used in this work over that of (Bongard 2008) is that a single CTRNN controller is evaluated in multiple environments in which the target object is placed at different locations. The final fitness of the controller is computed as

$$f = \min_{b=1}^{S} f_{sub}(b) \tag{2.4}$$

where S is the number of target object locations or sub-evaluations that the CTRNN is evaluated for and $f_{sub}(b)$ is the fitness of the CTRNN on sub-evaluation b (see eqns. 2.2,2.3). Using the minimum fitness over all sub-evaluations renders a given CTRNN only as fit as it is in its weakest sub-evaluation which prevents finding CTRNNs that specialize at picking up the target object in one location, but do not work well in others.

1. **IncrementalShaping**()
2. Create and evaluate random parent p
3. `WHILE` ∼Done()
4. Create child c from p, and evaluate
5. `IF` Fitness(c) $\geq$ Fitness(p) AND (PreviousSuccess(c) OR Success(c)) [see Eqns. 2.2,2.3,2.4]
6. $p = c$
7. `IF` Failure()
8. EaseEnvironment()
9. Re-evaluate p
10. `WHILE` Success(p)
11. HardenEnvironment()
12. Re-evaluate p

---

13. **Done**()
14. 30 hours of CPU time have elapsed

---

15. **Failure**()
16. 100 generations since last success

---

17. **EaseEnvironment**()
18. EvaluationTime ← EvaluationTime+100

---

19. **Success**(g)
20. $\exists k, k \in \{1, \ldots, t\}$ |
21. T(LeftgripperTip, k)&
22. T(RightgripperTip, k)&
23. D(SensorNode, k) ≥ 0.825

---

24. **PreviousSuccess**(g)
25. TargetDistance ← TargetDistance-0.01m
26. success = Success(g)
27. TargetDistance ← TargetDistance+0.01m
28. RETURN success;

---

29. **HardenEnvironment**()
30. TargetDistance ← TargetDistance+0.01m

Figure 2.2: **Incremental Shaping pseudocode.** The algorithm executes a hill climber [1-14] (see text for description). If the current genome fails [15,16], the task environment is eased [17,18]; while it is successful [19-23], the task environment is made more difficult [24,25]. $T(x, k)$ returns 1 if body part x is in contact with another object and zero otherwise at time step k. $D(x, k)$ returns the value of the distance sensor located at body part x at time step k.

A hill climber (Russell and Norvig 2002) is used to optimize the initial random CTRNN against this fitness function. At each generation a child CTRNN is created from the current best CTRNN and mutated. Mutation involves considering each τ, w, θ and n value in the child, and replacing it with a random value in

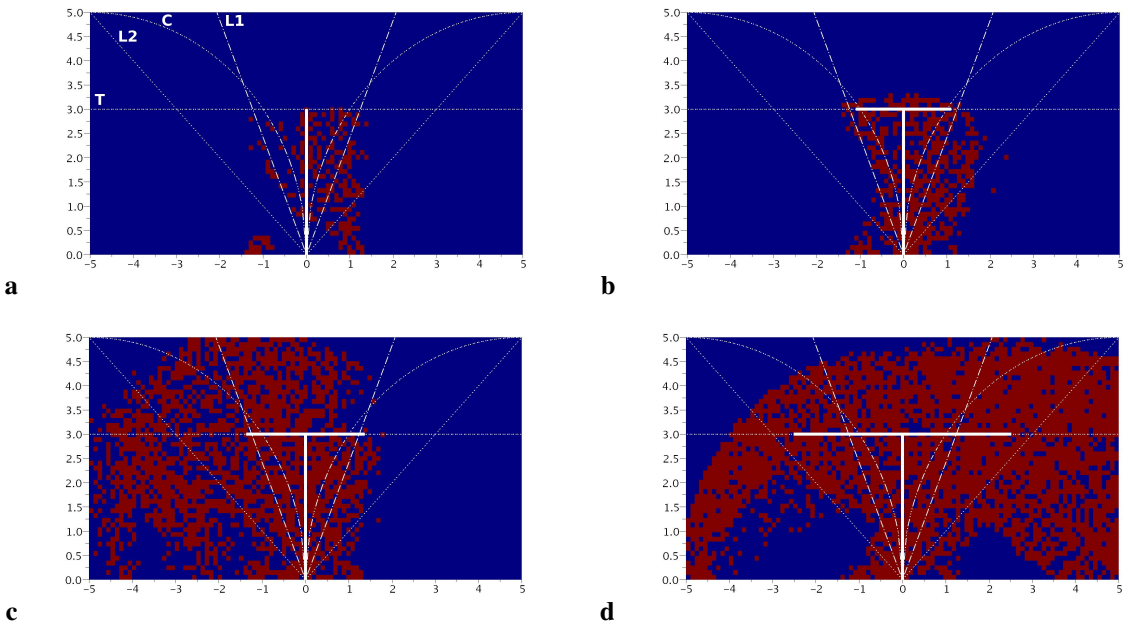


Figure 2.3: Sample generalization plots from evolution of a generalized controller on robot 2. Red indicates the robot was successful at picking up the target object at that location, while blue indicates the robot was unsuccessful at that location. The four scaffolding schedules are superimposed. Specifically the plots shown are for controllers that were successful at distances of 3 meters (**a**), 3.2 meters (**b**), 3.3 meters (**c**) and 3.92 (**d**) the final training distance reached in this run.

its range with a probability of $10/209 = 0.0478$. This ensures that, on average, 10 mutations are incorporated into the child according to a binomial distribution[3]. If the fitness of the child CTRNN is equal to or greater than the fitness of the current best CTRNN, and the child CTRNN is either successful at picking up the target object in either the current or previous environment, then the best CTRNN is replaced by the child; otherwise the child is discarded. This ensures that the grasping behavior learned in previous environments is retained while the locomotion behavior is adapted to the current environment.

After each possible replacement, the current CTRNN is considered in order to determine whether a failure condition has occurred, or whether it has achieved the success criteria. In the present work the failure condition is defined as 100 generations of the hill climber elapsing before a successful CTRNN is found. A successful CTRNN is defined as one for which, at some time step during the current evaluation both gripper

[3]The original publication of this paper stated that this was a normal distribution, which is incorrect.

tips touch the target object and it is lifted far enough onto the robot's back such that the distance sensor there fires above a certain threshold.

If the failure condition occurs, the task environment is eased; if the current CTRNN succeeds, the task environment is made more difficult. Easing the task environment involves increasing the current evaluation period by 10 time steps. This has the effect of giving the robot more time to succeed at the current task if it fails. Making the task environment more difficult involves moving the target object further away from the robot. This has the effect of teaching the robot to grasp and lift the target object when it is close, and learning to turn and locomote toward the target object, followed by grasping and lifting it, when it is placed further away. As some CTRNNs that succeeded for a given target object distance also succeed when the target object is moved further away, the target object is continually moved until the current CTRNN no longer succeeds, at which time hill climbing recommences. In order to further speed the algorithm an individual evaluation is terminated early if the robot ceases to move before succeeding at the task.

2.2.4 Scaffolding Schedules

As mentioned above each CTRNN is evaluated at multiple target object locations. These locations are a function of the distance of the target object from the robot, which increases with each success. Specifically, four different such functions, or scaffolding schedules were compared in this work. All four attempt to select first for grasping followed by a combination of turning and locomoting. The schedules are created in this way because it was shown in (Bongard 2008) that selecting for grasping first proved the best way to achieve both grasping and locomotion.

The first scaffolding schedule, henceforth referred to as '**T**', begins with only one sub-evaluation and places the target object in front of the robot at increasing distance until the target object is a distance of three meters from the robot. It was observed that by this distance, the robot must have learned a stable gait to reach the target object. As distance is increased past three meters the target object is moved out in both directions along the line perpendicular to the robot's sagittal plane, requiring two sub-evaluations: one sub-evaluation with the target object placed in front and to the left, and another in which the target object is placed in front

and to the right of the robot. Formally

$$(x, z) = \begin{cases} (0, L), & \text{if } L \leq 3.0 \\ (\pm\sqrt{L^2 - 9.0}, 3.0), & \text{otherwise} \end{cases} \tag{2.5}$$

where L is the distance of the target object from the robot's start location. This schedule is depicted graphically as the thick lines in Fig. 2.3. The next schedule used is

$$(x, z) = (\pm L^2/10.0, \sqrt{10.0|x| - x^2}) \tag{2.6}$$

that is the target object is moved concurrently along the perimeter of circles with radius 5 meters and centers at 5 and -5 meters ('**C**'). In this case two sub-evaluations are always used. The final two schedules both move the target object away from the robot linearly on both sides. One does so with a slope $m = 1/\tan(22.5°)$ ('**L1**') and the other does so with a slope $m = 1/\tan(45°) = 1$ ('**L2**'). In both these cases the function used is

$$(x, z) = (\pm L/\sqrt{(m^2) + 1}, |mx|) \tag{2.7}$$

See Fig. 2.3 for a graphical representation of these schedules.

In order to speed evaluation of child CTRNNs in schedules with multiple sub-evaluations, if the sub-fitness of the first sub-evaluation attempted by the child CTRNN is lower than the fitness of the current best CTRNN (which was set to its lowest scoring sub-fitness), then no additional sub-evaluations are performed and the child CTRNN is discarded.

2.2.5 Measuring Performance

In order to evaluate the quality of an evolved CTRNN, two metrics are considered. The first is how far away the target object was placed at the end of 30 hours of training. While this metric is useful for judging how rapidly the robot can adapt to a changing environment it does not measure how successful a given CTRNN is in unseen environments. For this purpose a generalization metric has been devised. If the point directly in front of the robot is considered to be the origin of a Euclidean space, then a 10 meter by 5 meter grid extending from $(-5, 0)$ to $(5, 5)$ can be constructed and a controller can be systematically tested to determine

1. **GeneralizationTest**()
2. NumSuccesses = 0;
3. FOR $x = -5; x \leq 5; x+ = 0.1$
4. FOR $z = 0; z \leq 5; z+ = 0.1$
5. Place target object at (x, z) and
   let simulation run for 10,000 time steps
6. If Success() [see Fig. 2.2]
7. NumSuccesses++;
8. RETURN (NumSuccesses / 5151)

Figure 2.4: Generalization Test Pseudocode. The 10x5 grid is uniformly sampled at 101x51=5151 target object locations to determine percentage of grid coordinates where the controller is successful.

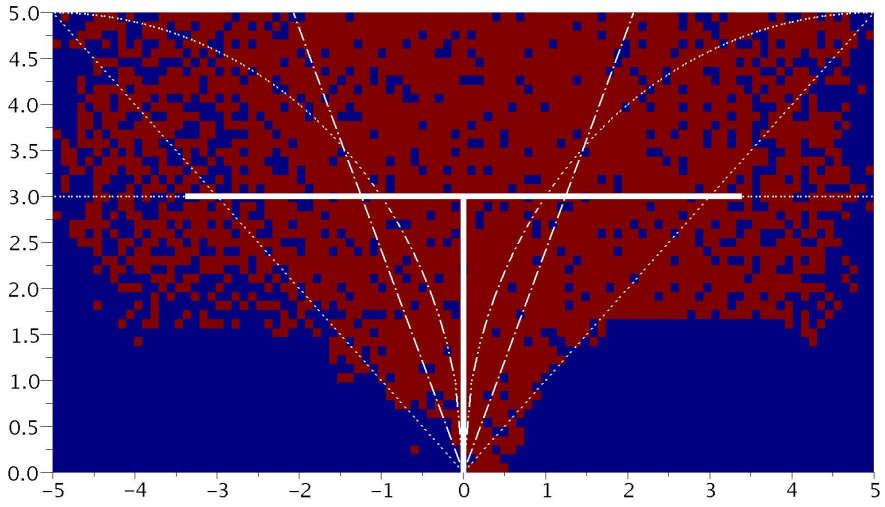


Figure 2.5: Generalization plot from best controller for robot 1.

how well it performs the task for a sampling of target object locations within this grid. Specifically, this grid is sampled as shown in Fig. 2.4. Additionally, for each grid position, whether or not the controller was successful there is recorded and can be plotted as shown in Figs. 2.3 and 2.5.

Table 2.1: The best generalization values of the final controllers from each experiment.

Schedule:	**T**	**C**	**L1**	**L2**
Robot 1:	53.6%	32.5%	23.3%	13.2%
	20.2%	28.3%	19.7%	12.7%
	16.6%	24.7%	14.9%	9.7%
	15.2%	24.3%	13.2%	9.2%
	15.1%	22.7%	11.5%	9.0%
Robot 2:	57.7%	26.3%	24.7%	12.6%
	40.4%	24.8%	24.1%	8.9%
	28.4%	21.4%	21.9%	7.6%
	27.4%	19.3%	19.6%	5.8%
	26.4%	19.1%	13.5%	4.8%

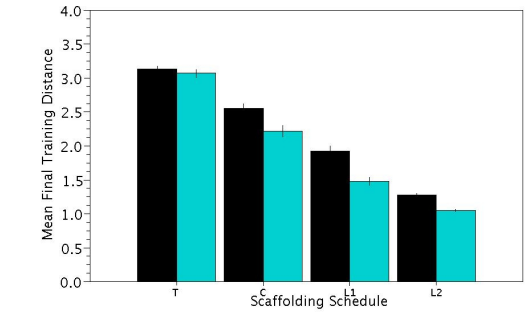
2.3 Results

For each robot and each scaffolding schedule mentioned above a set of 100 independent runs were conducted giving a total of $2*4*100 = 800$ total runs. Each run consisted of running the incremental shaping algorithm for 30 hours of CPU time. At the completion of each run, the generalization test as described in Fig. 2.4 was performed on the final CTRNN from that run to test its ability to generalize to unseen environments. For each set of runs, the mean final target object distance and the mean generalization percent of those final CTRNNs are plotted in Fig. 2.6. While the mean generalization score for each set of runs was under 10% in all instances, there were runs in each set that found controllers with much higher generalization values. The generalization scores for the final controllers from the top five runs from each set are given in Table 2.1.

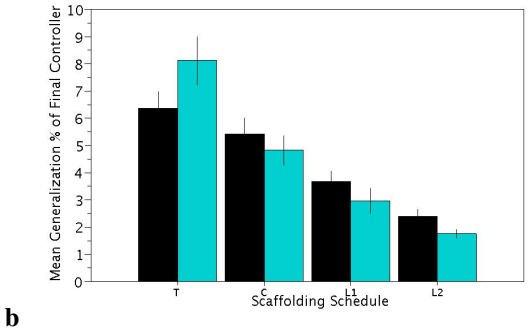
The **T** scaffolding schedule significantly outperforms the other three schedules both in training distance achieved and generalization, for both robots. Comparing performances between robots, it is noted that the **T** schedule evolves significantly more generalized controllers with the second robot (left hand grouping in Fig. 2.6b,c) while reaching similar final training distances as the first robot (left hand grouping in Fig. 2.6a). While the relative performance of the four schedules remains consistent across robots, the three other schedules lead to slightly less generalized controllers with the second robot (three right hand groupings in Fig. 2.6b).

2.3.1 A Sample Evolved Controller

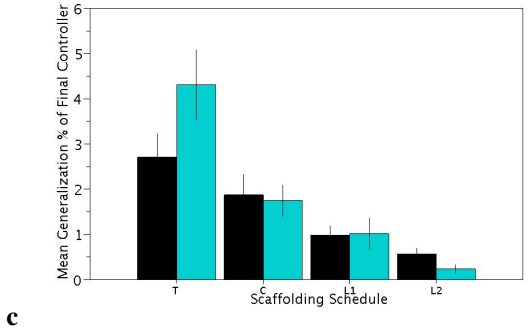
Fig. 2.7 shows the behavior of the controller that achieved the highest generalization score overall, which comes from using the **T** schedule with robot 2. Here it can be seen how the behaviors differ based on target



a



b



c

Figure 2.6: Performance of robots evolved under the different scaffolding schedules. Mean final distance achieved in training (**a**) mean generalization % of final CTRNN (**b**), and mean generalization % of final CTRNN for all locations where $x \notin [-1, 1]$ (**c**) across the 100 runs for each of the two virtual robots (robot 1 in black, robot 2 in blue) and each of the four scaffolding schedules. All plots include standard error bars.

object locations. Fig. 2.7a-h show the robot picking up the target object when it is located in front and to the right of the robot's initial position. The robot actually turns too far to the right while approaching the target object and then straightens itself out before picking up the target object. Fig. 2.7i-p show the same CTRNN controlling the robot to pick up the target object when it is located forward and to the left of the robot's initial

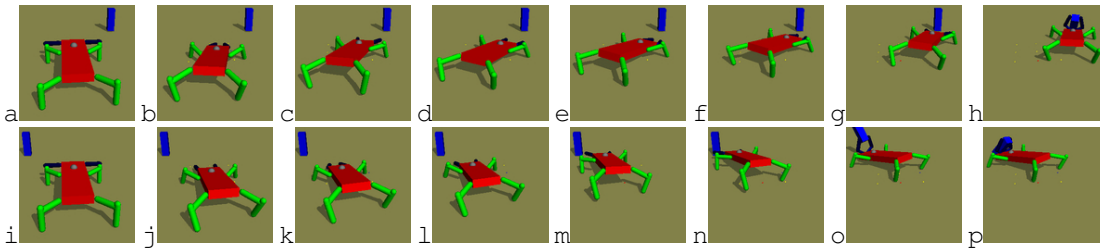


Figure 2.7: A sample successful controller for **robot 2**. **a-h**: The robot moves toward the target object placed 3.9 meters ahead and to the right while turning (**a-d**), turns too far (**e,f**), compensates (**g**) and then picks up the target object (**h**). **i-p**: The robot moves toward the target object placed equal-distance away on the left side without overshooting (**i-n**) and swings it onto its back (**o-p**).

position. In this case the robot does not turn too far, but approaches the target object at an angle that allows it to swing the target object onto its back.

The results of the generalization test performed on this same CTRNN are shown in Fig. 2.3d. This plot is colored red for all the locations where the CTRNN was successful in picking up the target object, and blue where it was not. This controller was able to pick up the target object in over 50% of target object locations. Specifically, there are large number of locations at which the CTRNN is successful even though it was never exposed to these locations during its training. Also it is noted that this CTRNN is successful for the majority of locations it would have experienced under any of the other scaffolding schedules, indicating it is possible for a controller to succeed at those locations, but that it can only do so after forward locomotion has been learned (as enforced by the **T** scaffolding schedule).

Fig. 2.3a-c show generalization plots for controllers from the same run as Fig. 2.3d that were saved when the robot was successful at training distances of 3, 3.2, and 3.3 meters respectively; that is, these controllers are ancestors of the final CTRNN from this run. It can be seen that there is a discontinuous jump in generalization between 3.2 and 3.3 meters. This illustrates how between these two distances, the increased pressure for the controller to learn turning resulted in a much greater ability to generalize to unseen environments once turning was mastered.

2.4 Discussion

2.4.1 Order Matters

The question presents itself as to why the **T** scaffolding schedule results in more successful controllers than any of the other schedules. The justification given in (Bongard 2008) is that the order in which the necessary behaviors needed to complete a task are selected for greatly affects the probability that all behaviors will be learned. In that work it was shown that if the robot was trained to pick up the target object first followed by training for locomotion it was more successful than if it was trained to locomote first and then trained to pick up the target object. Based on this result, all four schedules presented in this work select for grasping first, but the **T** schedule allows the robot to learn forward locomotion and then additionally learn the taxis behavior. The other three schedules each, to varying degrees, pressure the robot to learn turning toward the target object either before or while learning to locomote. This proves that, because these three schedules are less successful, forward locomotion should be learned before turning, for both robot morphologies. As can be seen in Fig. 2.6 the probability of training a controller to enable taxis and object manipulation is inversely proportional to the pressure to learn turning before locomotion: the **T**, **C**, **L1**, and **L2** schedules decline in performance, but increase in the pressure they exert to learn turning before locomotion.

2.4.2 Training Milestones

Another way to consider why the **T** schedule yields the most successful controllers is that it forces the evolved controllers to achieve certain milestones during training. Fig. 2.8 reports the rate at which both robots overcome these milestones using the **T** scaffolding schedule. Almost all runs rapidly reach around one meter, the furthest point at which the robot can pick up the target object by leaning or lunging forward without having to take any steps. The drop in learning rate (represented by increased slope) at this point denotes the difficulty in incorporating an oscillatory dynamic into the controller to allow stepping while retaining the dynamic that allows grasping and lifting once the target object is reached. This is the first learning milestone. Between one and three meters the learning rate is relatively constant: CTRNN parameters are tuned to enable stable oscillations, which induce rhythmic motion in the legs, thus carrying the robot to the target object. This is the second learning milestone.

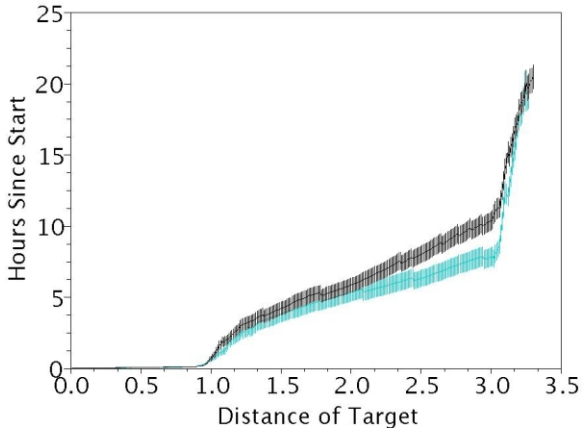


Figure 2.8: Target object distance vs. mean time to reach that distance. This plot shows for the **T** schedule: robot 1 (black) and robot 2 (blue), the mean time to reach each target object distance during training with standard error bars. All distances reached by at least 30 runs are included. Many runs surpassed this distance, but are not shown for the sake of clarity.

When the target object is placed more than three meters from the robot and an increasing distance away from its sagittal plane, there is a growing asymmetry in the distance sensor values reported by the two claw tips at the outset of an evaluation. This point corresponds to an apparent slowing in the learning rate as shown by the greater slope to the right in Fig. 2.8. It is acknowledged that the learning rate is expected to slow somewhat as the controller is now evaluated in two environments instead of one (the target object is placed to the right and then to the left). However, it can be seen that the learning rates for the two robots are not the same: robot 2 more rapidly adapts to target object placements further from its sagittal plane than robot 1 does. This indicates that the slowed learning rate is not only a result of the increased evaluations, but is also a function of morphology and behavior: robot 2's morphology eases the transition from forward locomotion to directed locomotion better than robot 1's morphology does.

2.4.3 Morphology Matters

Fig. 2.9 plots the maximum distance to which the target object was moved during training against the number of runs (out of 100) that produced successful controllers for that distance before their time limit of 30 hours expired. It can be seen that more of the runs using robot 1 discovered controllers that drove the robot to a distance of three meters, compared to the runs using robot 2 (the blue line is above the black line between one and three meters in Fig. 2.9). This is presumably due to the fact that controllers can be more easily trained

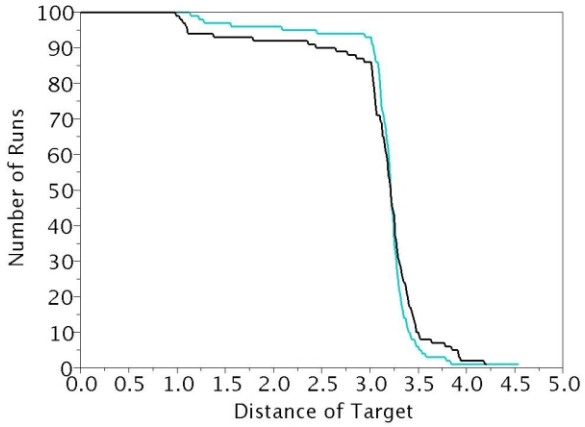


Figure 2.9: Target object distance vs. number of runs reaching that distance. This plot shows for the **T** schedule: robot 1 (black) and robot 2 (blue) the number of runs reaching each target object distance during training.

to produce forward locomotion in robot 1, which has legs parallel to its sagittal plane and therefore to its direction of travel. However, more runs using robot 2 discover controllers that allow the robot to reach and manipulate the target object when it is placed beyond three meters and away from its centerline, evidenced by the crossing of the lines around 3.2 meters. This is presumably due to the splayed legs of robot 2 allowing for directed locomotion more easily.

This observation is strengthened by Fig. 2.5, which reports the generalization ability of the best controller evolved for robot 1. Despite the robustness of this controller (it guides the robot toward success in 53.6% of the target object placements), the robot is rarely successful in regions that require a small turning radius (the two regions in the lower left and right of Fig. 2.5). This further suggests that robot 2 is better able than robot 1 to learn turning.

One last piece of evidence supporting this observation can be seen in Fig. 2.6c. Here the generalization abilities of the two robots across all four scaffolding schedules are compared, but these values are calculated considering only target object placements outside of $x \in [-1, 1]$: locations that require turning, because the target object is at least one meter away from the robot's sagittal plane. It is noted that the difference in scores between robot 1 and 2 using the **T** schedule are greater in this plot than in Fig. 2.6b, in which all target object locations are considered. This further confirms that controllers evolved for robot 2 are more likely to be able to pick up target objects at locations that require turning.

2.5 Conclusions and Future Work

This work has demonstrated that with the proper scaffolding schedule (**T**) it is possible to evolve controllers capable of performing a non-trivial sequence of behaviors even in previously unseen environments. Moreover it has demonstrated that altering morphology can impact the performance achievable through incremental shaping: robot 2 resulted in more generalized behaviors than robot 1.

However, for the two morphologies considered in this work it does not alter the sequence in which behaviors should be learned. Robot 2's splayed legs make turning easier, however scaffolding schedules that select for turning before locomotion is learned were not better able to integrate object manipulation, turning and locomotion into a controller using this body plan. Therefore it is concluded that the task environment, the learning algorithm, and/or the evolvability of CTRNNs dictate learning sequence more than morphology does.

In order to strengthen this conclusion more morphologies will need to be considered. Future work will investigate how additional morphologies perform under these scaffolding schedules. Additionally the authors intend to investigate how evolving the robot's body plan along with its controller may result in less sensitivity to the order in which behaviors are learned. This would simplify the application of shaping for realizing multiple dynamic behaviors in intelligent agents.

2.6 References

Beer, R. D. (2006). Parameter space structure of continuous-time recurrent neural networks. *Neural Comp. 18*(12), 3009–3051.

Beer, R. D. (2008). The dynamics of brain-body-environment systems: A status report. In P. Calvo and A. Gomila (Eds.), *Handbook of Cognitive Science: An Embodied Approach*, pp. 99–120. Elsevier.

Bongard, J. (2008). Behavior chaining: incremental behavioral integration for evolutionary robotics. In S. Bullock, J. Noble, R. Watson, and M. A. Bedau (Eds.), *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pp. 64–71. MIT Press, Cambridge, MA.

Brooks, R. (1986). A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of [legacy, pre - 1988] 2*(1), 14–23.

Calabretta, R., S. Nolfi, D. Parisi, and G. P. Wagner (2000). Duplication of modules facilitates the evolution of functional specialization. *Artificial Life 6*(1), 69–84.

Carriker, W., P. Khosla, and B. Krogh (1991). Path planning for mobile manipulators for multiple task execution. *IEEE Transactions on Robotics and Automation*, 403 – 408.

Chella, A., H. Dindo, F. Matraxia, and R. Pirrone (2007). Real-time visual grasp synthesis using genetic algorithms and neural networks. In R. Basili and M. T. Pazienza (Eds.), *AI*IA*, Volume 4733 of *Lecture Notes in Computer Science*, pp. 567–578. Springer.

Dorigo, M. and M. Colombetti (1994). Robot shaping: Developing situated agents through learning. *Artificial Intelligence 70*(2), 321–370.

Fernandez Jr., J. J. and I. D. Walker (1999). A biologically inspired fitness function for robotic grasping. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith (Eds.), *Proc. of the Genetic and Evolutionary Computation Conf. GECCO-99*, San Francisco, CA, pp. 1517–1522. Morgan Kaufmann.

Foss, J., F. Moss, and J. Milton (1997). Noise, multistability, and delayed recurrent loops. *Physical Review E 55*(4), 4536+.

Harvey, I., P. Husbands, D. Cliff, A. Thompson, and N. Jakobi (1997). Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems 20*, 205–224.

Hornby, G., S. Takamura, T. Yamamoto, and M. Fujita (2005). Autonomous evolution of dynamic gaits with two quadruped robots. *Robotics, IEEE Transactions on 21*(3), 402–410.

Inamura, T., I. Toshima, and H. Tanie (2004). Embodied symbol emergence based on mimesis theory. *International Journal of Robotics Research 23*(4), 363–377.

Izquierdo, E. and T. Buhrmann (2008). Analysis of a dynamical recurrent neural network evolved for two qualitatively different tasks: walking and chemotaxis. In S. Bullock, J. Noble, R. Watson, and M. A. Bedau (Eds.), *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pp. 257–264. MIT Press, Cambridge, MA.

Nolfi, S. and D. Floreano (2000). *Evolutionary Robotics: The Biology,Intelligence,and Technology*. Cambridge, MA, USA: MIT Press.

Okada, M. and Y. Nakamura (2004). Design of the continuous symbol space for the intelligent robots using the dynamics-based information processing. *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on 4*, 3201–3206 Vol.4.

Reil, T. and P. Husbands (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. *Evolutionary Computation, IEEE Transactions on 6*(2), 159–168.

Russell, S. J. and P. Norvig (2002). *Artificial Intelligence: A Modern Approach* (Second ed.). Prentice Hall.

Saksida, L., S. Raymond, and D. S. Touretzky (1997). Shaping robot behavior using principles from instrumental conditioning. *Robotics and Autonomous Systems 22*, 231–249.

Seraji, H. (1998). A unified approach to motion control of mobile manipulators. *The International Journal of Robotics Research 17*(2), 107–118.

Singh, S. P. (1992). Transfer of learning across sequential tasks. *Machine Learning 8*, 323–339.

Wood, D., J. Bruner, and G. Ross (1976). The role of tutoring in problem solving. *J Child Psychol Psychiatry 17*(2), 89–100.

Chapter 3

Evolution of Functional Specialization in a Morphologically Homogeneous Robot

A central tenet of embodied artificial intelligence is that intelligent behavior arises out of the coupled dynamics between an agent's body, brain and environment. It follows that the complexity of an agents's controller and morphology must match the complexity of a given task. However, more complex task environments require the agent to exhibit different behaviors, which raises the question as to how to distribute responsibility for these behaviors across the agents's controller and morphology. In this work a robot is trained to locomote and manipulate an object, but the assumption of functional specialization is relaxed: the robot has a segmented body plan in which the front segment may participate in locomotion and object manipulation, or it may specialize to only participate in object manipulation. In this way, selection pressure dictates the presence and degree of functional specialization rather than such specialization being enforced *a priori*. It is shown that for the given task, evolution tends to produce functionally specialized controllers, even though successful generalized controllers can also be evolved. Moreover, the robot's initial conditions and training order have little effect on the frequency of finding specialized controllers, while the inclusion of additional proprioceptive feedback increases this frequency.

3.1 Introduction

Proponents of embodied artificial intelligence argue that intelligent behavior arises out of the coupled dynamics between an agent's body, brain and environment (Brooks 1999, Anderson 2003, Pfeifer and Bongard 2006, Beer 2008). One corollary of this view is that the complexity of the agents's controller and morphology must match the complexity of the task at hand. However, more complex task environments require the agent to exhibit different behaviors, which raises the question as to how to distribute responsibility for these behaviors across the agents's controller and morphology. It has been argued (Brooks 1986, Calabretta et al. 2000) that controllers should be organized in a modular fashion such that different control components are responsible for different behaviors, but others have shown that such structural modularity is not always necessary (Bongard 2008, Izquierdo and Buhrmann 2008, Auerbach and Bongard 2009).

In addition to modularity in structure, modularity can be thought of in terms of the functions that an agent performs. Moreover, this separation of function can be 'proximal', that is as seen from the point of view of the system itself, i.e. a description from the point of view of a robot's sensory-motor system that accounts for how the agent reacts to different sensory stimulation. It can also be 'distal', i.e. a high level description from the point of view of an independent observer that describes the behavior of an entire sequence of sensory-motor steps (Calabretta et al. 1998).

When constructing a system to solve a given problem either through engineering or evolution a mapping is created from a functional space (objectives) to a physical space (how to achieve them). Specifically, the objectives are defined in terms of functional requirements in the functional space and the physical embodiment is defined in terms of design parameters in the physical space. A design is a mapping from the functional requirements to the design parameters. This mapping is not unique and often there are infinitely many viable solutions, but a specific solution is found through the creative process of a human engineer or through an automated process such as evolution (Suh 1990).

Partly due to the human bias that favors breaking a problem down into separable, simpler sub-problems, roboticists often implicitly design such mappings to be functionally modular in the 'distal' sense: different parts of the robot's body are responsible for different behaviors. For example, wheels or legs may allow for movement while a separate gripper allows for object manipulation. In this work we investigate a robot trained to locomote and manipulate an object, but in which this assumption of functional modularity or specialization

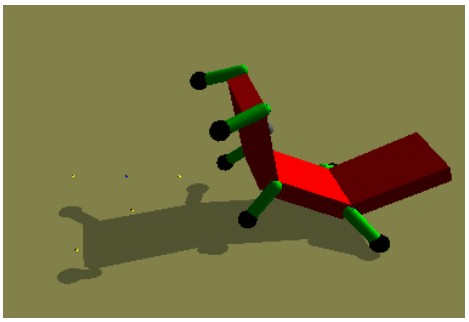


Figure 3.1: The virtual hexapod robot used in this work.

of different body parts is relaxed: the robot has a segmented body plan in which the front segment may participate in locomotion and object manipulation, or it may be specialized such that it only participates in object manipulation. In this way, selection pressure dictates the presence and degree of functional specialization rather than enforcing such specialization *a priori*.

In the next section the virtual robot and the incremental shaping method used for training the robot are introduced. The following section reports results demonstrating how changes in initial conditions, training order, and the inclusion of additional proprioceptive feedback affect the success of the evolved controllers and the frequency of evolution discovering functionally specialized controllers. The final section provides some discussion of the observed results, discusses multiple hypotheses that could explain the variability observed in the degree of specialization of evolved controllers across several different experimental regimes, and presents directions for potential future work.

3.2 Methods

This section first describes the virtual robot used in this work followed by a description of its controller. Next the incremental shaping algorithm used for training the robot is presented. The section concludes with a description of the metrics used to evaluate the evolved controllers.

3.2.1 The robot

In this work a virtual hexapod robot is used (Fig. 3.1). The robot is composed of three homogeneous body segments attached to each other with one degree of freedom joints that rotate through the robot's sagittal

plane. At the outset of an evaluation period, the segments are arranged horizontally (Fig.3.3a). The intersegmental joints may rotate neighboring segments toward one another up to $90°$. Two legs are attached to the anterior edge of each segment, one on each side. Each leg is attached to its segment with a universal joint that rotates through the sagittal plane with a range of $[-45°, 45°]$ and through the coronal plane with a range of $[-45°, 45°]$. A joint angle of $0°$ for both degrees of freedom maintains the leg perpendicular to its segment. Each leg is capped with a spherical foot.

Twelve motors actuate the six legs, and another two motors actuate the joints between body segments for a total of 14 motors. A touch sensor and distance sensor reside in each of the two front feet, and a distance sensor is embedded in the robot's back, for a total of five sensors. The touch sensors return a value of one when the corresponding body part touches another object and zero otherwise. The distance sensors return a value commensurate with the sensor's distance from the target object: they return zero if they are greater than five meters from the target object and a value near one when touching the target object. Object occlusion is not simulated here; the target object can be considered to be emitting a sound, and the distance sensors respond commensurately to volume.

The robot's controller is evolved such that the robot locomotes toward, grasps and lifts a rectangular target object placed in its environment.

3.2.2 The controller

The robot is controlled by a continuous time recurrent neural network (Beer 2006). The CTRNN is composed of eight motor neurons. Each pair of legs shares two motor neurons: one motor neuron controls rotation through the sagittal plane for both legs, while the other motor neuron controls rotation through the coronal plane for both legs. Sharing motor neurons ensures that when grasping the object the front legs close symmetrically, while also reducing the size of the controller and therefore the dimensionality of the search space. The remaining two motors control the joints between body segments and each receive commands from their own motor neuron. The value of each motor neuron is updated according to

$$\dot{y_i} = \frac{1}{\tau_i}\left(-y_i + \sum_{j=1}^{8} w_{ji}\sigma(y_j + \theta_i) + \sum_{j=1}^{5} n_{ji}s_j\right) \qquad \text{for } 1 \leq i \leq 8 \tag{3.1}$$

where y_i is the state of neuron i, w_{ji} is the weight of the connection from neuron j to neuron i, τ_i is the time constant of neuron i, θ_i is the bias of neuron i, n_{ji} is the weight of the connection from sensor j to neuron i, s_j is the value of sensor j and $\sigma(x) = 1/(1 + e^{-x})$ is the logistic activation function.

The virtual robot with a given CTRNN controller is evaluated over a set number of simulation steps in a physical simulator[1]. For each simulation step, using a step size of 0.0005, the sensors, CTRNN, joint torques and resulting motion of the robot are updated.

3.2.3 Training

The same incremental shaping (Singh 1992, Dorigo and Colombetti 1994, Saksida et al. 1997) algorithm presented in (Bongard 2008, Auerbach and Bongard 2009) is used for dynamically tuning the robot's task environment to facilitate learning. This method is outlined in Fig. 3.2. In short, the target object is initially placed in front of the robot such that it learns to grasp and lift the object. Once it does, the target object is moved slightly further away from the robot and training recommences. This process is repeated such that the robot must eventually learn locomotion as well as object manipulation in order to grasp and lift distantly-located objects.

More specifically, a random CTRNN is initially created by choosing all τ from the range [0.1, 0.5], all w from [-16, 16], all θ from [-1, 1], and all n from [-16, 16]; these ranges were found useful in previous work (Bongard 2008). This gives a total of $8 + 8 * 8 + 8 + 5 * 8 = 120$ evolvable parameters. The robot is then equipped with this controller and allowed to behave in a task environment for 100 time steps in which the target object is placed directly in front of the robot. After evaluation the fitness of the controller is computed as

$$f = \begin{cases} \max_{k=1}^{t}(D(\text{LFF}, k) * D(\text{RFF}, k)), & \text{if } !g(k) \\ 1 + \max_{k=1}^{t}(H(\text{TarObj}, k)), & \text{if } g(k) \end{cases} \tag{3.2}$$

where t is the number of time steps during the evaluation, $T(x, k)$ indicates that the touch sensor in body part x fired during time step k, $D(x, k)$ returns the value of the distance sensor in body part x during time step k, and $H(\text{TarObj}, k)$ indicates the height of the target object from the ground plane. The fitness awarded is

[1] Open Dynamics Engine: www.opende.org

therefore conditional on whether the robot has successfully grasped the object, which is defined as

$$g(k) = \ (T(\text{LFF}, k) == 1) \text{ AND } (T(\text{RFF}, k) == 1) \text{ AND } \\ (D(\text{LFF}, k) > 0.89) \text{ AND } (D(\text{RFF}, k) > 0.89) \tag{3.3}$$

which ensures grasping is only indicated when both touch sensors in the front feet fire during some time step in the evaluation period, and that both distance sensors in the front feet are sufficiently close to the target object during the same time step. This latter condition allows the robot to distinguish between touching the ground with both feet and touching the object.

If the robot has not yet learned to grasp the object, the upper condition in Eqn. 3.2 determines fitness, which rewards the robot for minimizing the distance between its front feet and the object. Once it learns to successfully grasp the object the lower condition in Eqn. 3.2 determines fitness, which rewards the robot for lifting the object as high as possible.

A hill climber (Russell and Norvig 2002) is used to optimize the initial random CTRNN against this fitness function. At each generation a child CTRNN is created from the current best CTRNN and mutated. Mutation involves considering each τ, w, θ and n value in the child, and replacing it with a random value in its range with a probability of $10/120 = 0.0833$. This ensures that, on average, 10 mutations are incorporated into the child according to a binomial distribution. If the fitness of the child CTRNN is equal to or greater than the fitness of the current best CTRNN, and the child CTRNN is either successful at picking up the target object in either the current or previous environment, then the best CTRNN is replaced by the child; otherwise the child is discarded. This ensures that the grasping behavior learned in previous environments is retained while the locomotion behavior is adapted to the current environment.

After each possible replacement, the current CTRNN is considered in order to determine whether a failure condition has occurred, or whether it has achieved the success criteria. In the present work the failure condition is defined as 100 generations of the hill climber elapsing before a successful CTRNN is found. A successful CTRNN is defined as one for which, at some time step during the current evaluation both front feet touch the target object and it is lifted off the ground above a certain threshold.

If the failure condition occurs, the task environment is eased; if the current CTRNN succeeds, the task environment is made more difficult. Easing the task environment involves increasing the current evaluation

1. **IncrementalShaping**()
2. Create and evaluate random parent p
3. `WHILE` ∼Done()
4. Create child c from p, and evaluate
5. `IF` Fitness(c) $\geq$ Fitness(p) AND (PreviousSuccess(c) OR Success(c)) [see Eqns. 3.2,3.3]
6. $p = c$
7. `IF` Failure()
8. EaseEnvironment()
9. Re-evaluate p
10. `WHILE` Success(p)
11. HardenEnvironment()
12. Re-evaluate p

---

13. **Done**()
14. 30 hours of CPU time have elapsed `OR` TargetDistance $>$ 10m

---

15. **Failure**()
16. 100 generations since last success

---

17. **EaseEnvironment**()
18. EvaluationTime ← EvaluationTime+100

---

19. **Success**(g)
20. $\exists k, k \in \{1, \ldots, t\}$ |
21. T(LeftFrontFoot, k)`AND`
22. T(RightFrontFoot, k)`AND`
23. (min(D(LeftFrontFoot, k), D(RightFrontFoot, k)) ≥ 0.89)`AND`
24. H(TargetObject, k) > 1.5

---

25. **PreviousSuccess**(g)
26. TargetDistance ← TargetDistance-0.01m
27. success = Success(g)
28. TargetDistance ← TargetDistance+0.01m
29. RETURN success;

---

30. **HardenEnvironment**()
31. TargetDistance ← TargetDistance+0.01m

Figure 3.2: **Incremental shaping pseudocode.** The algorithm executes a hill climber [1-14] (see text for description). If the current genome fails [15,16], the task environment is eased [17,18]; while it is successful [19-24], the task environment is made more difficult [30,31]. $T(x, k)$ returns 1 if body part x is in contact with another object and zero otherwise at time step k. $D(x, k)$ returns the value of the distance sensor located at body part x at time step k. $H(x, k)$ returns the height of object x at time step k

period by 10 time steps. This has the effect of giving the robot more time to succeed at the current task if it fails. Making the task environment more difficult involves moving the target object further away from

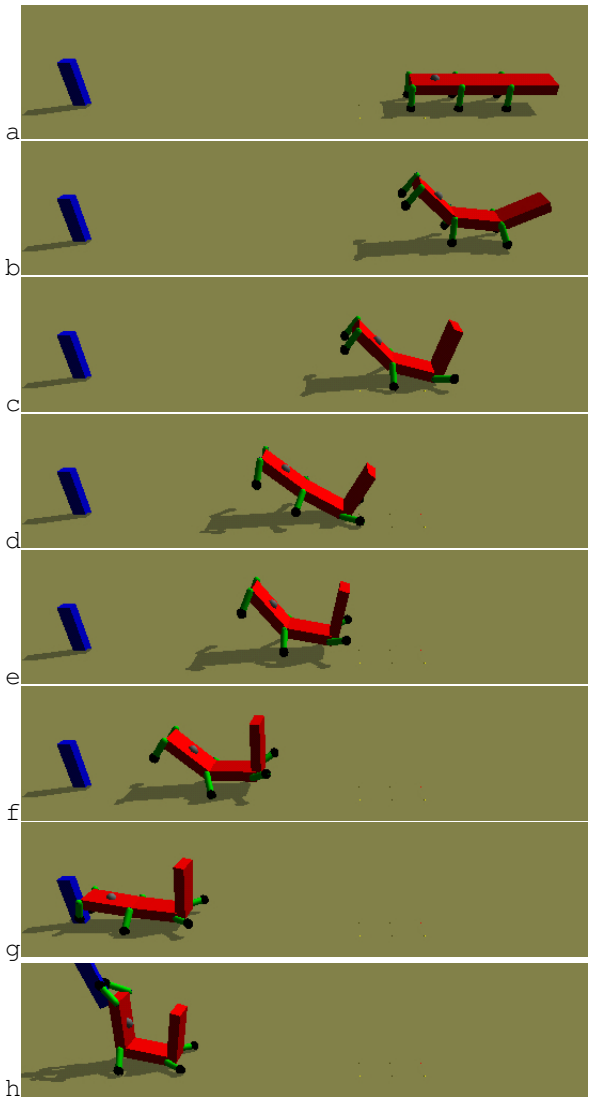


Figure 3.3: Sample functionally specialized controller. The robot's front body segment is raised and the front feet are kept off the ground during locomotion, i.e. they are only used for grasping the target object.

the robot. This has the effect of teaching the robot to grasp and lift the target object when it is close, and learning to locomote toward the target object, followed by grasping and lifting it, when it is placed further away. As some CTRNNs that succeeded for a given target object distance also succeed when the target object is moved further away, the target object is continually moved until the current CTRNN no longer succeeds, at which time hill climbing recommences. In order to further speed the algorithm an individual evaluation is terminated early if the robot ceases to move before succeeding at the task.

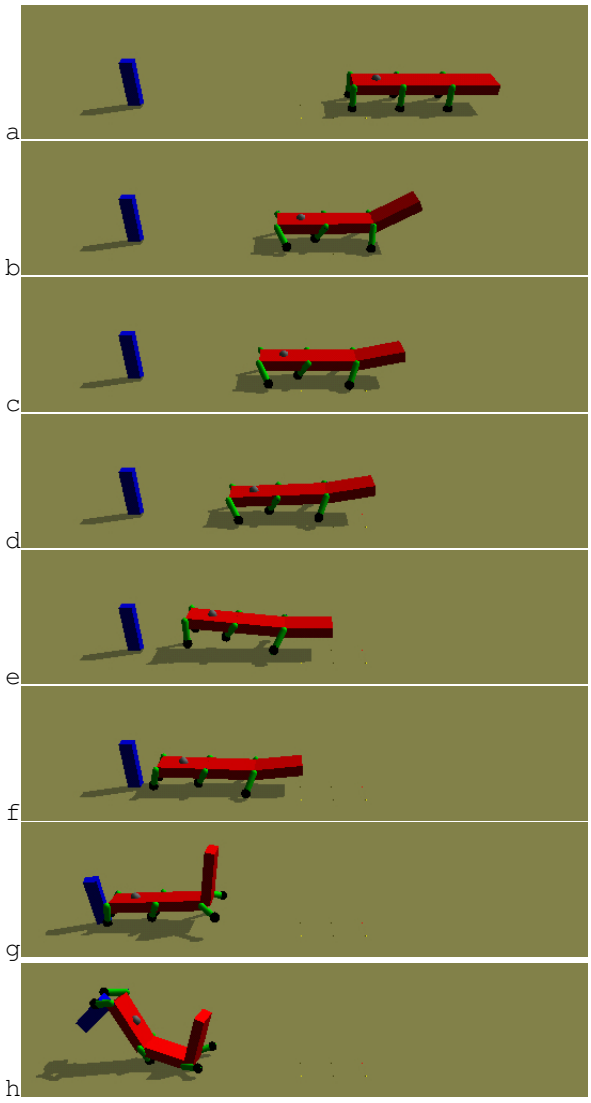


Figure 3.4: Sample functionally generalized controller. This controller uses the robot's front legs for propulsion during locomotion and for grasping and lifting of the target object.

3.2.4 Evaluating functional specialization

The two main questions of interest in the current work are (1) whether a single CTRNN acting as a monolithic controller for this robot can evolve to successfully locomote toward, grasp and lift the target object, and (2) if so whether the evolved controllers are functionally specialized in the 'distal' sense or not. To answer the first question it is sufficient to consider the distance of the target object from the robot at the end of training.

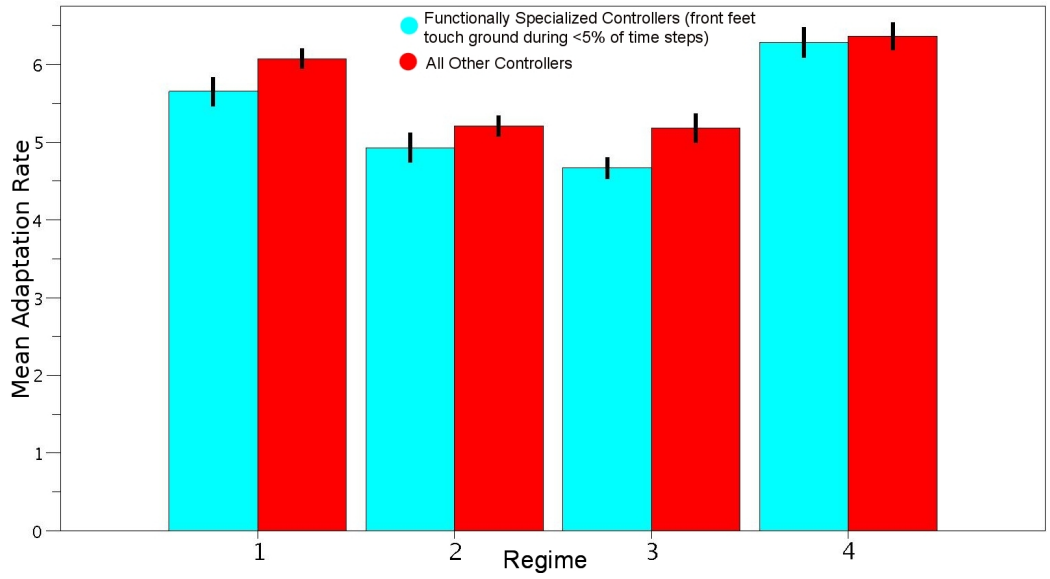


Figure 3.5: Plot of mean adaptation rate by regime. Errors bars depict one unit of standard error. Data is split between those controllers that cause the robot's feet to touch the ground during less than 5% of time steps (leftmost grouping in Fig. 3.6) and all others.

The greater this distance, the more simulations were performed in which the robot was considered to be successful, and the more rapidly the controller was able to adapt to changing environmental conditions. This metric will be referred to as the **adaptation rate**.

To investigate the second question one must consider that the robot's serially homogeneous body plan was designed such that it may locomote using all six legs or, alternatively, may rotate the anterior (or posterior) segment upward to locomote using the four middle and posterior (or anterior and middle) legs. A controller may then involve the front legs in both locomotion and grasping by keeping the front segment horizontal or, alternatively, it may restrict the front legs such that they only contribute to object manipulation. These latter controllers would realize functional specialization if locomotion and object manipulation are considered as two separate functions.

In order to evaluate whether a given successful controller is functionally specialized or not the simulation is run until the controller grasps the target object, while recording the sensor values during each time step. At the completion of this simulation the percent of total time steps during which both front feet touch sensors fire is calculated. Controllers with low values for this metric are considered to be functionally specialized because the robot rarely touches its front feet to the ground during locomotion. Conversely, controllers that use their

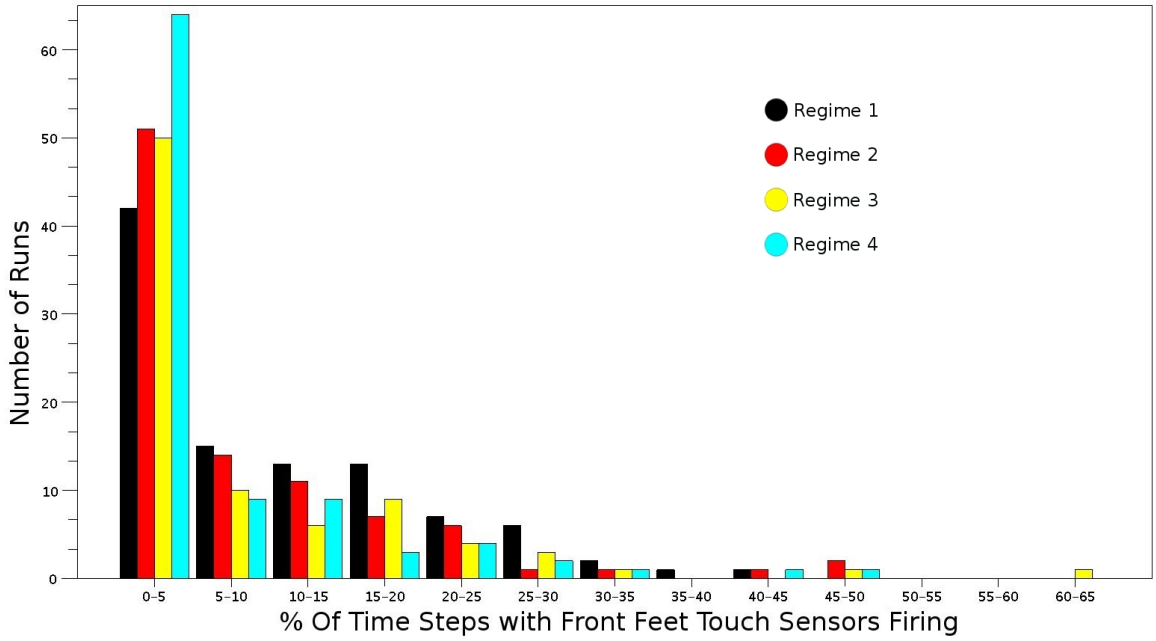


Figure 3.6: Histogram of the specialization metric for each of the four regimes. All runs in which the target object reached at least three meters are included (100 runs from regime 1, 94 runs from regime 2, 85 runs from regime 3, and 94 runs from regime 4).

front feet both for locomotion (either for propulsion, balance or both) and grasping are not functionally specialized and will receive higher values from this test. See Fig. 3.3 for an example of a functionally specialized controller (feet only touch in 0.076% of time steps), and Fig. 3.4 for an example of a functionally generalized controller (feet touch in 48.693% of time steps).

3.3 Results

Using the above methods four different experimental regimes were investigated and their results compared. Each regime consisted of running 100 independent trials of the incremental shaping algorithm (Fig. 3.2) with identical initial environmental conditions but different randomly-generated controllers. In the first experiment (**regime 1**) the front body segment joint was rotated upward 90° such that it was perpendicular to the ground with the front feet pointing forward and the target object was initially placed directly in front of the robot. All of the runs from this regime can be considered successful in the sense that they were able to adapt to target objects placed at distances greater than three meters (a distance that requires locomotion), grasp, and lift up

the object (see Fig. 3.5). Additionally, many of the runs from this regime resulted in functionally specialized controllers (black bars in Fig. 3.6).

In the second regime (**regime 2**) the robot was initialized with both body segments horizontal so that all six feet started on the ground and again the target object was initially placed directly in front of the robot. It was assumed that starting the robot flat would bias evolution to initially discover and retain locomotion involving all six legs, and therefore not specialize the front legs only for grasping. However, while still finding successful controllers in the majority of trials, the number of controllers resulting from this regime that developed functionally specialized controllers was similar to regime 1, and in fact counter to intuition more controllers from this regime caused the robot to touch their feet to the ground in less than 5% of time steps as compared with regime 1 (red bars in Fig. 3.6).

In the third experiment (**regime 3**) the body segments started horizontally, but in this case the target object was initially placed two meters away from the robot, so that before learning to grasp the target object the robot would first be forced to learn to move toward it. Without initial evolutionary pressure to involve the front legs in grasping it was assumed that the controllers to evolve in this experiment would be more likely to include them in locomotion, but once again a similar number of controllers resulting from this experiment developed functionally specialized controllers as compared with regimes 1 and 2 (yellow bars in Fig. 3.6).

The fourth regime (**regime 4**) was identical to regime 2 in that the body segments were started parallel to the ground with the target object initially directly in front of the robot. However, for this experiment two additional sensors were added to the robot: joint angle sensors for the two joints connecting the body segments, and these were wired to the controller. The controllers that evolved in this regime not only performed better in the sense that they adapted more rapidly to changes in the target object's position during training as compared to regime 2 (Fig. 3.7), but also were more likely to be functionally specialized when compared to the other three regimes (blue bars in Fig. 3.6).

3.4 Discussion and Conclusions

After noting that all four regimes were able to successfully learn both locomotion and object manipulation in the majority of trials the question arises as to why evolution tends to converge on functionally specialized

behaviors, and why the inclusion of additional sensors causes an increase in the frequency of converging on such behaviors. Three possible hypotheses are: (1) functionally specialized controllers are more evolvable, and therefore supplant less specialized controllers during an evolutionary run, (2) evolution initially discovers a specialized or generalized controller, and subsequently improves on that behavior but does not increase or decrease specialization, and (3) functionally specialized behaviors more easily allow for active perception (Noë 2004).

Hypothesis (1) is supported by previous work, which has indicated that modularity can increase evolvability (Wagner and Altenberg 1996), but only under certain environmental conditions (Kashtan and Alon 2005, Lipson et al. 2002). However, Fig. 3.5 indicates that for two of the four regimes (regimes 2 and 4) studied here, adaptation rate is similar between those runs that converged on functionally specialized behaviors and those that converged on generalized behaviors, and in fact adaptation rate was lower within runs containing specialists compared to generalists in the other two regimes (regimes 1 and 3). This suggests that functionally specialized behaviors do not arise because they are more evolvable, but for some other reason.

Hypothesis (2) suggests that evolution may become 'locked in' to a specialized or generalized strategy, depending on which type it discovers at the outset: it may be difficult to subsequently evolve the robot's controller to selectively tune the amount of behavioral specialization of one part of the body. It follows from this that the amount of specialization may be biased by the initial conditions of the robot during shaping. If scaffolding teaches grasping before locomotion or, more strongly, begins with the front segment raised vertically, controllers may converge on behaviors that allow the front legs to grasp the object, but evolution cannot subsequently co-opt those legs to participate in locomotion as well. However, this hypothesis is contradicted by Fig. 3.5, which indicates that changing the initial conditions to favor usage of the front legs in locomotion (regimes 2 and 3) do not produce more generalized controllers: these regimes also converge in the majority of runs on functionally specialized controllers. Hypothesis (2) is further invalidated by the run illustrated in Fig. 3.8, which shows that evolution may in some cases co-opt the front legs for increased participation in locomotion.

According to hypothesis (3), it may be that the robot is better able to actively perceive the proximity of the object—and therefore determine desirable conditions for lifting—if the front legs do not participate in locomotion, because then the touch sensors will only fire when in contact with the target object. Such controllers may be easier for the evolutionary process to find and optimize. Indeed, it has been demonstrated

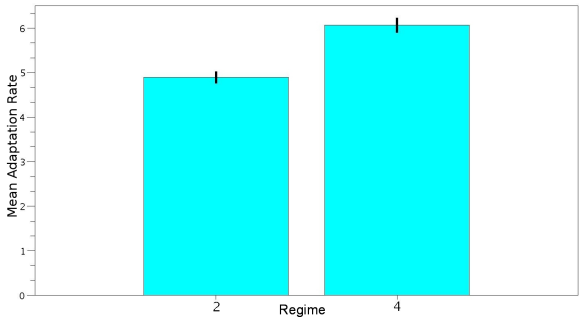


Figure 3.7: Plot of mean adaptation rate with standard error bars for regimes 2 and 4.

in the literature that active categorical perception may evolve in learning agents (Beer 2003). Moreover, providing the robot with additional proprioceptive feedback in regime 4 not only increased the prevalence of functional specialization (as shown in Fig. 3.6), but also the adaptation rate within those runs that produced specialized controllers (as shown in Fig. 3.7). It is plausible that these added sensors allow for better active perception as the touch sensors and sensed body posture may together indicate appropriate conditions for object manipulation.

Several additional experiments were designed to test this hypothesis. These experiments followed the theme of regimes 2 and 4. Specifically, in all cases the body segments were started parallel to the ground with the target object initially directly in front of the robot. What varied in these experiments were the sensors the robot was equipped with. Since a variable number of sensors results in a variable number of parameters under evolutionary control these experiments all used a fixed mutation rate of $\frac{10}{288} \approx 0.035$. Experiment **a** used the same sensors as regime 1 above, these sensors will be referred to as the base sensor set. Experiment **b** used the sensors of regime 4: the base sensor set with two joint angle sensors on the two joints connecting the main body segments added in. Experiment **c** used a robot with the base sensor set plus two more joint angle sensors: one apiece for the two degrees of freedom of the front left leg (just the left leg was used, because due to the construction of the controller the left and right legs operated symmetrically). Experiment **d** used a robot with the base sensor set plus two additional joint angle sensors on the middle left leg, and similarly experiment **e** used a robot with the base sensor set plus two additional joint angle sensors on the rear left leg. Experiment **f** used a robot with the base sensor set plus all the joint angle sensors featured in experiments **b-d**, while experiment **g** used a robot with the base sensor set plus touch sensors on the rear four feet. Experiment **h** used a robot with the base sensor set plus distance sensors on the rear four feet, and finally experiment **i**

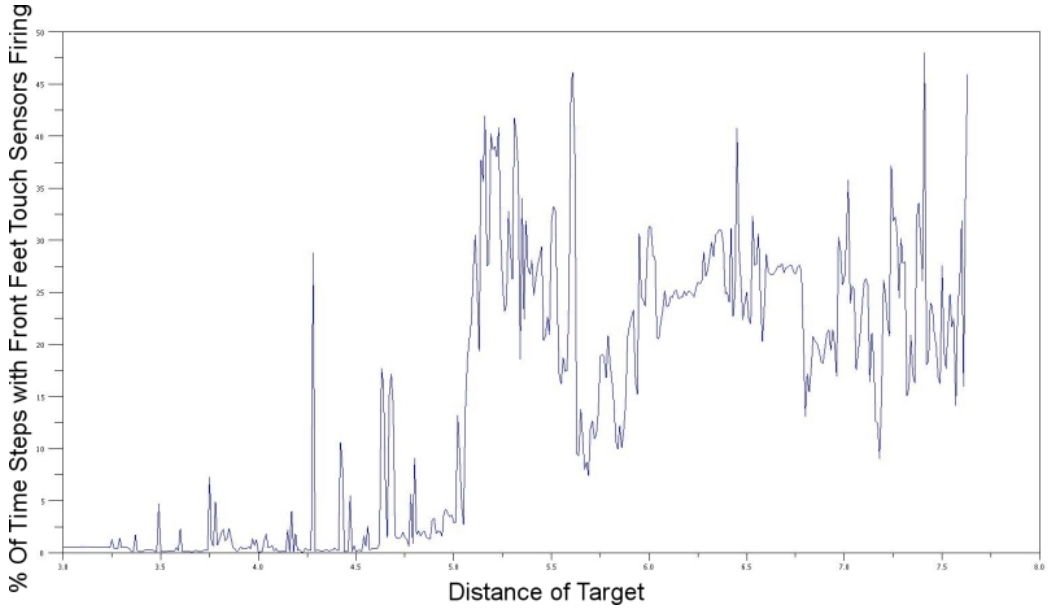


Figure 3.8: Evolution co-opting the front legs for increased participation in locomotion. Plot depicts the target object distance where controller was successful vs. % of time steps with front feet touch sensors firing from a single evolutionary run.

used a robot with all the sensors of experiment **f** plus the additional touch sensors and distance sensors on the rear four feet used in **g** and **h**.

Fig. 3.9 shows the mean adaptation rates with standard error bars for each of these additional experiments. Note the steady decline in performance from experiment **b** through experiment **e**. This result provides further evidence for hypothesis (3) as it demonstrates that adaptation rate declines as the included sensors provide less information in regards to desirable conditions for lifting: the main body joints (**b**) are most informative as discussed above, while the front leg angles may provide some information about the relative position of the front feet. As the sensors are moved toward the rear of the body less of this relevant information is available. This is further demonstrated by experiment **f** which shows that including all of the joint angle sensors buys the robot very little above just including the most useful pair (**b**). Additionally it is seen from experiment **g** that additional touch sensors improve performance even more so than any angle sensors do, because touch sensors provide the most direct evidence as to which feet are on the ground and/or touching the target object.

To verify that the additional sensors provide relevant information useful for the current task and do not merely aid in locomotion, virtual robots were instantiated with the sensor configurations of experiments **b-e**

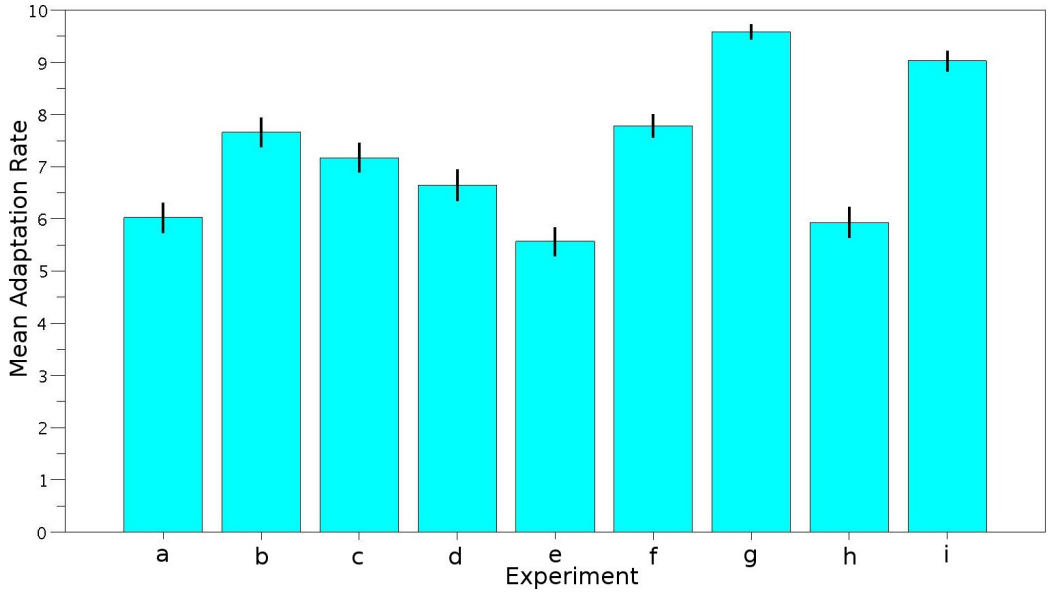


Figure 3.9: Mean adaptation rate with standard errors for additional experiments. See text for details.

and were evolved for locomotion alone. This consisted of expanding the range of the robot's distance sensors and placing the target object a large (100 m) distance away. Fitness was calculated as the fraction of distance between the start location and the target object location that the robot was able to cover in a set amount of time. Fig. 3.10 shows the mean fitnesses along with standard error bars from these experiments grouped by sensor configuration. Note that while including the joint angle sensors on the joints connecting the main body segments (**b**) leads to improved locomotion performance, there is no significant difference between the performance of the other three sensor sets. This provides further evidence that the differences observed across these configurations above are due to active perception.

In conclusion, it was shown here that evolution can tune the amount of functional specialization of different parts of the body. In future work we plan to evolve morphology as well as control: it is predicted that evolution would then specialize both the morphology and function for different body parts as the task environment dictates. This may prove to be a more fruitful method for realizing robots capable of an increasing number of behaviors, rather than fixing the body plan and manually assigning function to structure.

3.5 References

Anderson, M. (2003). Embodied Cognition: A field guide. *Artificial Intelligence 149*(1), 91–130.

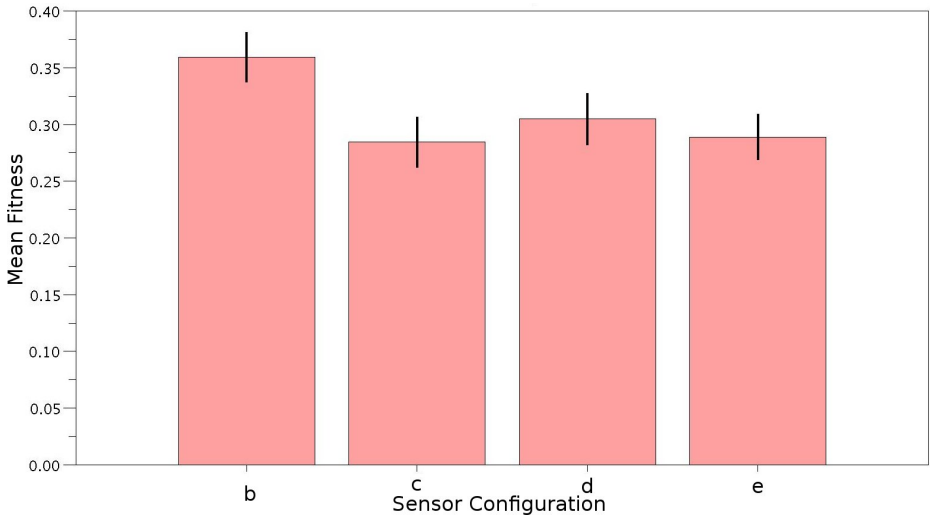


Figure 3.10: The impact of joint angle sensors on locomotion performance. Mean fitness with standard errors when selecting for just locomotion with the four different pairs of joint angle sensors.

Auerbach, J. and J. C. Bongard (2009). How robot morphology and training order affect the learning of multiple behaviors. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*.

Beer, R. (2003). The dynamics of active categorical perception in an evolved model agent. *Adaptive Behavior 11*(4), 209.

Beer, R. D. (2006). Parameter space structure of continuous-time recurrent neural networks. *Neural Comp. 18*(12), 3009–3051.

Beer, R. D. (2008). The dynamics of brain-body-environment systems: A status report. In P. Calvo and A. Gomila (Eds.), *Handbook of Cognitive Science: An Embodied Approach*, pp. 99–120. Elsevier.

Bongard, J. (2008). Behavior chaining: incremental behavioral integration for evolutionary robotics. In S. Bullock, J. Noble, R. Watson, and M. A. Bedau (Eds.), *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pp. 64–71. MIT Press, Cambridge, MA.

Brooks, R. (1986). A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of [legacy, pre - 1988] 2*(1), 14–23.

Brooks, R. (1999). *Cambrian intelligence*. MIT Press Cambridge, Mass.

Calabretta, R., S. Nolfi, D. Parisi, and G. P. Wagner (1998). Emergence of functional modularity in robots. In *Proceedings of the fifth international conference on simulation of adaptive behavior on From animals to animats 5*, Cambridge, MA, USA, pp. 497–504. MIT Press.

Calabretta, R., S. Nolfi, D. Parisi, and G. P. Wagner (2000). Duplication of modules facilitates the evolution of functional specialization. *Artificial Life 6*(1), 69–84.

Dorigo, M. and M. Colombetti (1994). Robot shaping: Developing situated agents through learning. *Artificial Intelligence 70*(2), 321–370.

Izquierdo, E. and T. Buhrmann (2008). Analysis of a dynamical recurrent neural network evolved for two qualitatively different tasks: walking and chemotaxis. In S. Bullock, J. Noble, R. Watson, and M. A.

Bedau (Eds.), *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pp. 257–264. MIT Press, Cambridge, MA.

Kashtan, N. and U. Alon (2005). Spontaneous evolution of modularity and network motifs. *Proc Natl Acad Sci U S A*.

Lipson, H., J. Pollack, and N. Suh (2002). On The Origin of Modular Variation. *Evolution 56*(8), 1549–1556.

Noë, A. (2004). *Action In Perception*. MIT Press.

Pfeifer, R. and J. Bongard (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press.

Russell, S. J. and P. Norvig (2002). *Artificial Intelligence: A Modern Approach* (Second ed.). Prentice Hall.

Saksida, L., S. Raymond, and D. S. Touretzky (1997). Shaping robot behavior using principles from instrumental conditioning. *Robotics and Autonomous Systems 22*, 231–249.

Singh, S. P. (1992). Transfer of learning across sequential tasks. *Machine Learning 8*, 323–339.

Suh, N. P. (1990). *The Principles of Design (Oxford Series on Advanced Manufacturing)*. Oxford University Press.

Wagner, G. and L. Altenberg (1996). Perspective: Complex adaptations and the evolution of evolvability. *Evolution 50*(3), 967–976.

Chapter 4

Evolving CPPNs to Grow Three-Dimensional Physical Structures

The majority of work in the field of evolutionary robotics concerns itself with evolving control strategies for human designed or bio-mimicked robot morphologies. However, there are reasons why co-evolving morphology along with control may provide a better path towards realizing intelligent agents. Towards this goal, a novel method for evolving three-dimensional physical structures using CPPN-NEAT is introduced that is capable of producing artifacts that capture the non-obvious yet close relationship between function and physical structure. Moreover, it is shown how more fit solutions can be achieved with less computational effort by using growth and environmental CPPN input parameters as well as incremental changes in resolution.

4.1 Introduction

Robots that operate in outdoor or other unstructured environments such as the home or office would be of great social utility. But, to date, the vast majority of robots currently in use operate only in structured environments such as factories. If robots are to make the migration from factories into our everyday lives they will need to be adaptive; that is, they must exhibit intelligent behavior.

According to proponents of embodied artificial intelligence such intelligent behavior arises out of the coupled dynamics between an agent's body, brain and environment (Brooks 1999, Anderson 2003, Pfeifer and Bongard 2006, Beer 2008). One extension of this concept is that the complexity of an agent's controller and morphology must match the complexity of the task or tasks that it is required to perform. However, when extending this idea to more complex agents in more complex environments it is not clear how to distribute responsibility for different behaviors across the agent's controller and morphology. For example, if all a robot needs to do is follow a light source over flat terrain wheels and a direct sensory motor mapping would be an appropriate solution (Braitenberg 1986), but if the robot must be able to navigate over a variety of terrains and perform more complicated tasks a more complex control strategy and/or morphology are required. This issue of scaling has been one of the major obstacles in developing robots capable of robust and adaptive behavior in unstructured environments.

4.1.1 Background

Evolutionary robotics (Harvey et al. 1997, Nolfi and Floreano 2000), in which evolutionary algorithms are employed to optimize the control policy of a robot, has provided one framework for overcoming the limitations of human intuition in designing robust, non-linear, control strategies. However, the majority of work in evolutionary robotics has done only that: optimize control strategy for a human designed or bio-mimicked robot morphology. This methodology has severe limitations: fixing a robot's morphology places limits and biases on the kinds of action that the robot can perform, and therefore also on the more complex behaviors that those actions may eventually support. For example, a robot with legs can only exhibit legged locomotion while a wheeled robot with a rigid gripper can only move over even terrain and grasp objects with a fixed radius.

However, there are ways to overcome these limitations. Evolutionary algorithms may be used to optimize robot morphology as well as the control policy. Sims (Sims 1994) first introduced an evolutionary framework in which both the morphology and control of simulated machines were optimized in virtual environments to produce adaptive behavior. This work was followed by other studies (Dellaert and Beer 1994, Lund et al. 1997, Adamatzky et al. 2000, Mautner and Belew 2000, Lipson and Pollack 2000, Hornby and Pollack 2001a, Hornby and Pollack 2001b, Stanley and Miikkulainen 2003, Eggenberger 1997, Bongard and Pfeifer 2001, Bongard 2002, Bongard and Pfeifer 2003) in which aspects of the machine's morphology and control

were evolved in virtual environments. This approach has the advantage of discovering body plans appropriate for the machine's task environment rather than being artifacts of human design biases or copies of animal body plans only appropriate for that animal's ecological niche.

In fact, it is sometimes possible to construct morphologies ideally suited to their task environment such that no active control is necessary to accomplish non-trivial behaviors. One class of such morphologies are passive dynamic walkers (Collins et al. 2005, Tedrake et al. 2005). These "robots" are able to stably locomote down an inclined plane without using any sensors or motors. Their stability and momentum conservation are inherent properties of their body plans. If the prevailing view held by robotocists that an appropriate morphology will always be intuitive to design, as summed up by Nelson *et al*.

> *Humans are much better at designing physical systems than they are at designing intelligent control systems: complex powered machinery has been in existence for over 150 years, whereas it is safe to say that no truly intelligent autonomous machine has ever been built by a human (Nelson et al. 2009, p. 22).*

were correct then it would be intuitive how to design such structures. However, such body plans are non-intuitive to design because they have subtle dependencies on mass distribution and structural curvatures, making automated methods including evolutionary algorithms good candidates for designing these artifacts.

In order to capture the essence of evolving non-trivial physical structures the current work tackles the problem of evolving solid objects with these important properties. While not incorporating any actuation and therefore not robots, strictly speaking, the structures evolved in this work provide a stepping stone for the eventual evolution of fully articulated robots controlled by closed loop, neural network control policies. This approach is not without precedent in the field of evolutionary robotics. Funes and Pollack (Funes and Pollack 1997) first demonstrated evolving solid structures such as bridges, scaffolds, and crane arms constructed of Lego bricks before moving on to the evolution of actuated robots (Pollack et al. 2001).

Specifically, the work presented in this paper makes use of a recently introduced abstraction of development known as compositional pattern producing networks (CPPNs) (Stanley 2007) to grow three-dimensional physical structures. In this paper, these CPPNs are evolved using CPPN-NEAT (Stanley 2007) to produce physical structures capable of conserving momentum to achieve maximum displacement due to gravity. CPPNs are used here because they are a form of indirect encoding that have been shown able to capture geometric symmetries appropriate to the system being evolved, are capable of reproducing outputs at multi-

ple resolutions (Stanley et al. 2009), and have shown promise in producing neural network control policies for legged robots (Clune et al. 2009, Clune et al. 2009). The combination of these features makes it likely that evolving CPPNs will prove to be a more promising approach to realizing intelligent agents than other approaches.

This paper presents a method for using CPPN-NEAT along with a novel growth procedure to evolve three dimensional structures appropriate for a specific task, presents some advantages of CPPN-NEAT over other methods that may be used for evolving three-dimensional physical structures, and discusses how this method can be extended to co-evolve actuated body plans and control strategies.

4.2 Methods

This section presents a brief description of CPPNs and the CPPN-NEAT evolutionary algorithm. This is followed by a description of the methods used for generating three-dimensional physical structures from evolved genotypes. Following this a description is presented of the fitness function used for evaluating these structures.

4.2.1 CPPNs

Compositional Pattern Producing Networks (CPPNs) are a form of artificial neural network (ANN) where each internal node can have an activation function drawn from a diverse set of functions instead of being limited to a sigmoid function as is the case with classical ANNs. This function set includes functions that are repetitive such as sine or cosine as well as symmetric functions such as gaussian, thus easily allowing for motifs seen in natural systems: symmetry, repetition, and repetition with variation.

4.2.2 CPPN-NEAT

CPPN-NEAT uses the NeuroEvolution of Augmenting
Topologies (NEAT) (Stanley and Miikkulainen 2002) method of neuro-evolution to evolve increasingly complex CPPNs. An extension of CPPN-NEAT —HyperNEAT— has been used (Stanley et al. 2009, Clune et al. 2009, Clune et al. 2009) to evolve traditional ANNs, where each node is embedded in a geometric space and whose coordinates are fed to an evolved CPPN. In effect the connections are “painted” on to the network

from the output patterns produced by the CPPN. As shown by Stanley *et al.* (Stanley et al. 2009) this has the crucial benefit that CPPNs evolved to produce the connectivity patterns of small ANNs can be re-queried at a higher resolution to produce the connectivity patterns of larger ANNs without needing to re-evolve these large ANNs. Analogously CPPNs evolved to produce structures at one resolution should be able to also produce structures at a higher resolution.

CPPNs have several properties desirable for generating robot morphologies. It is directly evident that geometry is a key aspect of any artifact existing in a physical or simulated physical environment. Providing the evolutionary process with information about this geometry should be useful in evolving functional structures. Additionally the ability to operate at multiple resolutions should allow for the rapid evolution of coarse grained structures composed of a small number of large components followed by re-querying the generating CPPN to produce qualitatively similar structures composed of a greater number of smaller components without needing to evolve CPPNs for these higher resolution morphologies from scratch.

Importantly, if the evolved structures are to be physically fabricated using a technique like that outlined in (Lipson and Pollack 2000) the structures' dynamical properties must be retained across the simulation-reality gap (Beer 1990, Watson et al. 1999). The ability to query the same evolved encoding at different resolutions should be useful in this endeavor since the dimensions and densities with which the evolved structures will be fabricated most likely will not be able to precisely match those of the simulation.

4.2.3 Growing Three-Dimensional Physical Structures from CPPNs

In this work three-dimensional physical structures are grown from evolved CPPNs. Each structure is composed of many spherical cells which fuse together to make rigid bodies. For an example of a structure produced in this way see Fig. 4.1.

The growth procedure begins with a single cell, henceforth referred to as the root, located at a designated origin. A cloud composed of n points is cast around this cell with the n points being evenly distributed on the surface of the root sphere (all n points are at distance r from the center of the root). This point cloud is cast using a spiral method that is a variant of the algorithm presented by Saff and Kuijlaars (Saff and Kuijlaars 1997). Specifically the method is the "golden section" modification to the Saff and Kuijlaars algorithm described in (CGAFaq 2010). Once this cloud is cast, every point in the cloud is used to query a CPPN to retrieve a single output value. This output value can be thought of as a concentration of matter at

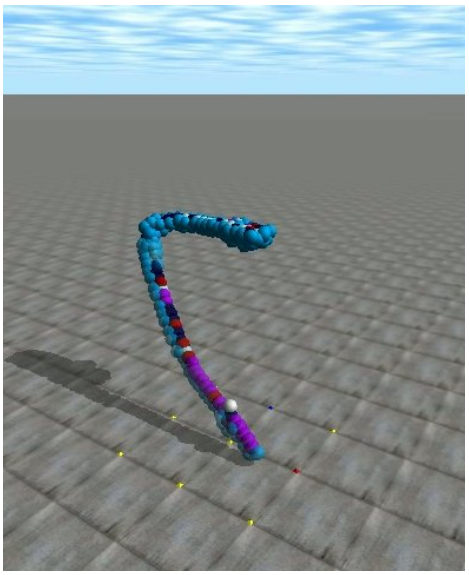


Figure 4.1: A sample structure evolved for maximum displacement due to gravity.

that point, such that when over a certain matter threshold, T_{matter}, a cell will be placed at that point. The more the output value exceeds the matter threshold the denser the cell placed at that point will be. This creates a continuum from no cell to very light cells to heavier cells (since the cells are all the same size, density and weight are completely correlated).

The CPPN takes as input the Cartesian coordinates (x, y, z) of the point in question as well as a constant bias input. Additionally information is provided as input about the growth trajectory itself: two angles ϕ_1 and θ_1 describe the direction from the parent cell that this point is located and an additional two angles (ϕ_2 and θ_2) provide directional information about the parent cell relative to its own parent cell. Further knowledge of the growth trajectory is provided via an input that receives the number of cells in the growth tree separating this point from the root (the cell's depth, d). Additional input is provided by a value representing the radius of the cells being considered for addition (r) thus informing the CPPN about the resolution at which the structure is being grown. Resolution will be dependent on the environment in which the physical structure exists, therefore this value may be considered an environmental input.

Once the output values for all n points in the cloud have been computed the points are sorted in order of descending output values. The sorted points are then looped through and the algorithm considers adding a cell centered at each point in turn. Specifically a cell, centered at point p is added to the structure if (a) the

```
1.  GrowStructure(CPPN)
2.      Initialize priority queue q, with priority based on
            cell density
3.      Create cell c at origin with full density, add to
            structure S and flag its coordinates 'discovered'
4.      Enqueue c in q
5.      WHILE ~ q.isEmpty
6.          c ← q.front
7.          Cast point cloud C centered at c
                using the golden section spiral method
8.          Initialize vector V of neighboring cells
9.          FOR EACH point p in C
10.             Query CPPN at p to get output value v
11.             Add p with value v to vector V
12.         Sort V by descending value
13.         FOR EACH point p with value v in sorted vector V
14.             IF coordinates of p not yet 'discovered'
15.                 Flag p 'discovered'
16.                 IF CanAdd(p,v,c)
17.                     Add cell centered at p with density
                            proportional to v to structure S
18.                     Enqueue (p, v) in q
-------------------------------------------------------------
19. CanAdd(p, v, c)
20.     IF v > T_matter AND
            ∀ cells d ∈ S, d ≠ c dist(p,d) ≥ r AND
            p is within bounding cube AND
            S has less than M cells
21.         Return true
22.     ELSE
23.         Return false
```

Figure 4.2: **Grow Structure pseudo code.** The growth procedure starts with a root node at the origin (line 3). Then, as long as there are nodes in the queue to consider it takes the node at the front of the queue, casts a point cloud around it and considers adding a node at each point in turn (lines 5-18). A node is added at a given point if all of the following hold: it does not conflict with a previously added node, the CPPN outputs a value above the threshold T_{matter} when queried at that point, the point is within the bounding cube, and the maximum number of nodes M has not been reached (lines 19-23).

output value of point p is above the threshold T_{matter} and (b) no other cell, besides the one to which this new cell will be attached (its parent) has previously been added to the structure with center located at distance $< r$ away from p.

When a cell is added to the structure it gets placed into a priority queue whose priority is based on the output value of the CPPN at that point. When all points from the current cloud have been considered the algorithm takes the cell at the top of the priority queue and casts a point cloud around it, and this process continues either until there are no possible points at which to place cells or a maximum number of cells (M) have been created. One further constraint is that the structure is not allowed to grow outside of a bounding cube with side lengths l. This constraint was imposed so that in the future it will be possible to physically

fabricate the entire evolved physical structures within the confines of a 3D-printer. Fig. 4.2 gives pseudo code for this growth procedure.

There are several reasons why it is desirable to have a growth procedure such as this. Merely querying CPPNs over a sampling of three-dimensional space may lead to disconnected objects. Even if all but one of these objects are thrown out much computational resources will have been wasted querying these regions of space. Additionally imposing a grid over space to determine which points to query restricts the sort of structures that may be produced when compared to the "point cloud" method discussed above. For example curved structures can be constructed using the point cloud method, but only coarsely approximated on a grid. Additionally, having a growth procedure allows for providing the CPPN with knowledge about the structure's growth trajectory and environment which prove to be beneficial (see below) and which may more easily allow for environmental influences to act on this growth trajectory in more complex ways in the future.

4.2.4 Selecting for dynamical properties of evolved structures

One major purpose of this paper is to demonstrate that CPPN-NEAT coupled with the growth procedure just presented is capable of evolving three-dimensional physical structures with desirable dynamic properties. This is a necessary capability of any evolutionary algorithm to be used for co-evolving robot morphology and control. In particular the property selected for in this work is the maximum displacement of an object due to gravity from a starting position where part of the object begins in contact with the ground.

To select for this property, an evolved virtual object is placed in a physical simulator[1] for a set amount of time. The fitness of this object is then calculated by finding the point of the object nearest to the origin of the space (where the object was touching the ground) in terms of the planar Euclidean distance. This distance becomes the fitness of the object and therefore of the CPPN that produced the object, which CPPN-NEAT attempts to maximize.

4.3 Results and Discussion

This section presents the results of several experiments designed to illustrate the capabilities of this methodology as well as study the effects of including the growth and environmental inputs introduced above and

[1]Simulations are conducted in the Open Dynamics Engine (`http://www.ode.org`), a widely used open source, physically realistic, simulation environment

Table 4.1: Summary of the parameters used in the different experiments. The full set of CPPN inputs includes the growth and environmental inputs while the restricted set does not.

	CPPN Input Set	**Resolution: first 100 generations**	**Resolution: second 100 generations**
Experiment 1	Full	$r = 0.1$m, $M = 200$	$r = 0.1$m, $M = 200$
Experiment 2	Restricted	$r = 0.1$m, $M = 200$	$r = 0.1$m, $M = 200$
Experiment 3	Full	$r = 0.15$m, $M = 60$	$r = 0.1$m, $M = 200$
Experiment 4	Restricted	$r = 0.15$m, $M = 60$	$r = 0.1$m, $M = 200$
Experiment 5	Full	$r = 0.08$m, $M = 391$	$r = 0.08$m, $M = 391$
Experiment 6	Restricted	$r = 0.08$m, $M = 391$	$r = 0.08$m, $M = 391$
Experiment 7	Full	$r = 0.1$m, $M = 200$	$r = 0.08$m, $M = 391$
Experiment 8	Restricted	$r = 0.1$m, $M = 200$	$r = 0.08$m, $M = 391$

the effects of varying the resolution at which structures are grown within a single evolutionary trial. All experiments are conducted with the fitness function described above and each experiment involves running 30 independent evolutionary trials. In all cases CPPN-NEAT is configured to use a population size of 150, and run for 200 generations. Additionally in all experiments the value T_{matter} is fixed at 0.7, and each cell of the structure is restricted to having its center within the bounding cube $([-2, 2], [-2, 2], [0, 4])$ (coordinates all in meters). The CPPN internal nodes are allowed to use the signed cosine, gaussian, and sigmoid activation functions. All other parameters of the evolutionary algorithm are kept at the default values provided with the C++ implementation of HyperNEAT[2].

In the first experiment (**experiment 1**) the radius r of each cell is fixed at 0.1 meters and the growth procedure is limited to $M = 200$ cells. **Experiment 2** is identical to experiment 1 except the growth and environmental CPPN inputs ϕ_1, θ_1, ϕ_2, θ_2, r and d are omitted so that just the basic Cartesian coordinates (x, y, z) along with the constant bias are inputted to each CPPN. Henceforth the inputs used in experiment 1 will be referred to as the full set, while those used in experiment 2 will be referred to as the restricted set.

Experiment 3 uses the full set of CPPN inputs and begins with the cell radius r fixed at 0.15 meters and with $M = 60$ cells maximum per structure. These parameter values remain fixed for the first 100 generations of each evolutionary trial. After the 100th generation the resolution of the structures is increased to that of the first two experiments: $r = 0.1$ meters, $M = 200$. These values, as well as those used in all subsequent changes of resolution are chosen to preserve the volume of the structures. As the radius is changed by a factor

[2]Available at `http://eplex.cs.ucf.edu/hyperNEATpage/HyperNEAT.html`

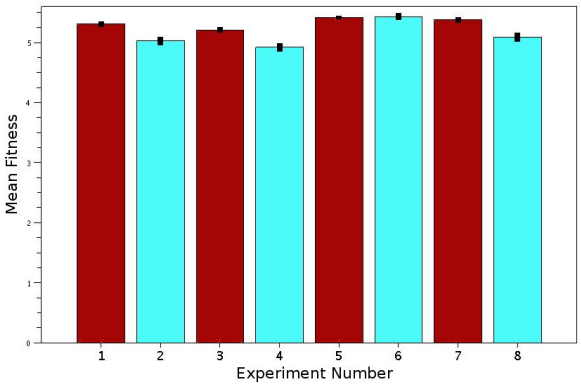


Figure 4.3: Mean best fitnesses in final generation. Mean displacements in meters taken across the 30 independent evolutionary trials are plotted with standard error bars for each of the eight experiments. The red bars represent experiments using the full set of CPPN inputs, while the blue bars represent experiments using the restricted set.

x the maximum number of cells is changed by a factor x^3. The next experiment, **Experiment 4**, is identical to experiment 3, except the CPPNs are limited to the restricted set of inputs.

Experiment 5, like experiment 1, uses the full set of CPPN inputs and keeps the resolution fixed for the duration of each evolutionary trial. However, in experiment 5 this resolution is higher. The cell radius r is set to 0.08 meters with a maximum, $M = 391$ cells per structure. **Experiment 6** is identical to experiment 5, but uses the restricted set of CPPN inputs.

Two final experiments, **experiments 7 and 8** follow the template of experiments 3 and 4. In both these experiments the resolution is increased halfway through each evolutionary trial, but in this case the resolution starts at $r = 0.1$ meters, $M = 200$ and increases to $r = 0.08$ meters, $M = 391$ after 100 generations. Once again these experiments are identical, beside experiment 7 using the full set of CPPN inputs while experiment 8 uses the restricted set. Table 4.1 summarizes all these experimental setups.

After completion of all evolutionary trials, statistics are computed for each experimental setup. Specifically of interest is how fit the evolved structures are, how long it takes for each evolutionary trial to complete, and how robust evolved CPPNs are to changes in the resolution at which they are queried.

Fig. 4.3 shows the mean best fitnesses from the final generation for each of the eight experiments. The first thing to notice in this figure is that in three of the four pairs of experiments the structures evolved using the full set of CPPN inputs (shown in red) on average achieve significantly higher fitness when compared with the structures evolved in the equivalent experiment using the restricted set. Moreover in the fourth pair

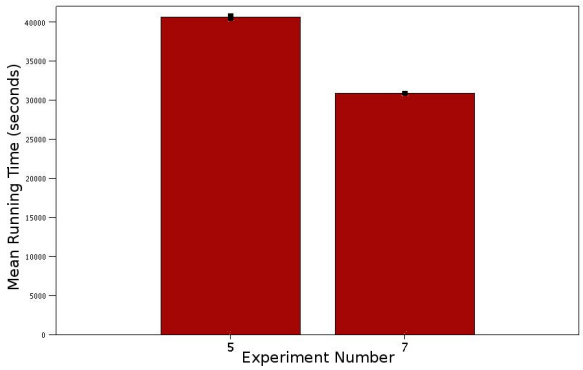


Figure 4.4: Mean running time in seconds to evolve structures at the highest resolution. Means are plotted with standard error bars for the two experiments that evolve structures at the highest resolution with the full set of CPPN inputs. Evolutionary trials in experiment 5 which always evaluate structures at the highest resolution take significantly longer than those of experiment 7 which evaluate structures at a reduced resolution for the first 100 generations.

of experiments (experiments 5 and 6) on average there is no significant difference in performance between those evolved with the full set of CPPN inputs and those evolved with the restricted set. This means that including the additional inputs never degrades performance of the evolved structures and more often than not leads to an improvement in performance.

Also noteworthy in this figure is that, when using the full set of CPPN inputs, on average there is no significant difference in the best fitness in the final generation when comparing between experiments that always evaluate structures at the highest resolution and those that spend the first half of their generations evaluating structures at a lower resolution (comparing experiment 1 with experiment 3 and experiment 5 with experiment 7). This is important, because as shown in Fig. 4.4 the evolutionary trials that spend their first 100 generations evaluating structures at a lower resolution run in significantly less time than those that always evaluate structures at the full resolution. This makes intuitive sense, because evaluating at the lower resolution requires fewer queries of the CPPN and allows for faster physical simulations due to the lower complexity of the structure being evaluated.

The final property of interest is how robust the evolved CPPNs are to changes in resolution. A key benefit of CPPN-NEAT over other evolutionary methods is that the evolved encoding are capable of producing structures at different resolutions, and as mentioned above if these structures are going to be physically fabricated it is important that they not be too sensitive to changes in resolution. Fig. 4.5 demonstrates how some of the evolved structures have similar dynamics and achieve similar performance when grown at

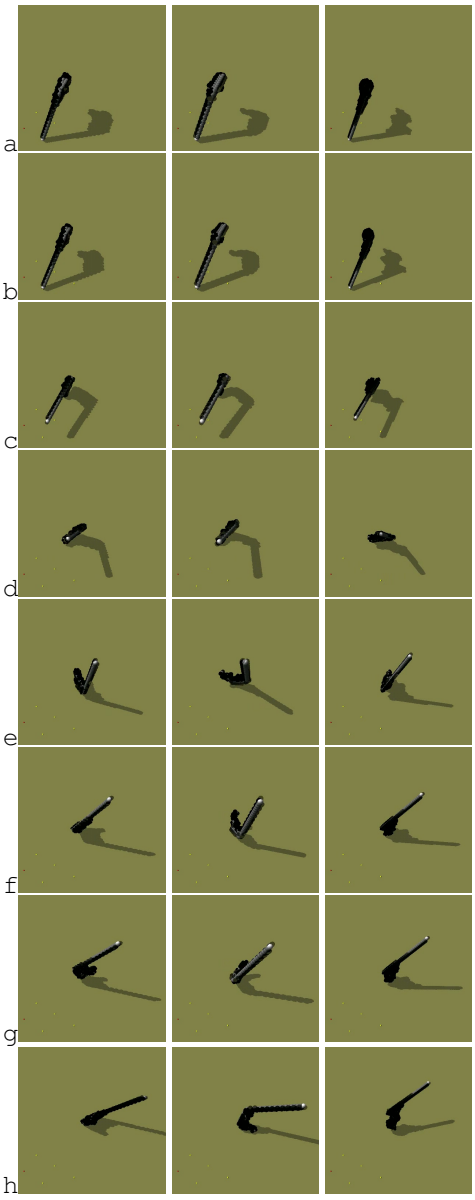


Figure 4.5: An evolved structure behaving similarly when grown at different resolutions. Left: Behavior of an evolved structure where each cell has radius $r = 0.08$ meters and is alloted a maximum $M = 391$ cells. Center: The CPPN used to generate the structure on the left is re-queried to produce a new structure at lower resolution: $r = 0.1$m, $M = 200$. Right: The same CPPN is re-queried again to produce a new structure at higher resolution: $r = 0.07$m, $M = 583$. Note how the three structures achieve similar performance even though structures were only evaluated at the $r = 0.08$m resolution during evolution. See Fig 4.7 for an enlarged view of these structures.

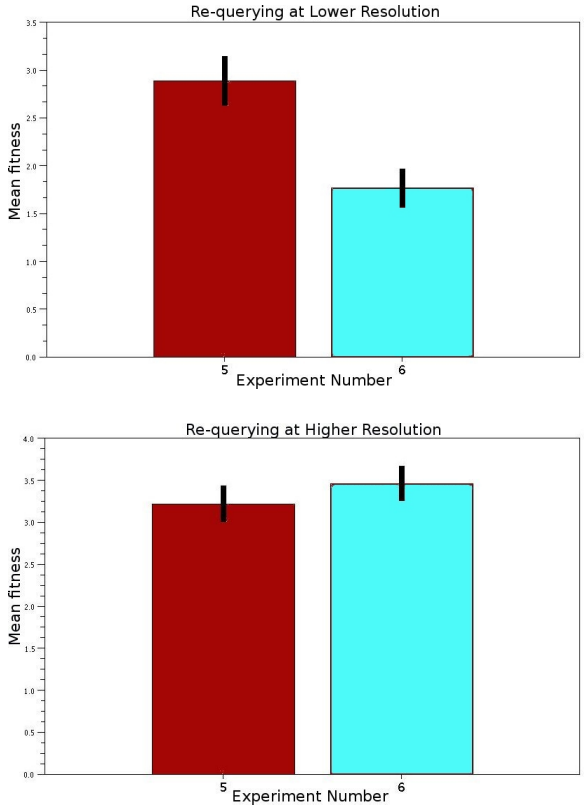


Figure 4.6: Comparing the performance of structures at different resolutions. Top: Mean fitnesses of structures grown from CPPNs evolved in experiments 5 and 6 by re-querying at a lower resolution ($r = 0.1$m, $M = 200$) with standard error bars shown. Bottom: mean fitnesses of structures grown from the same CPPNs by re-querying at a higher resolution ($r = 0.07$m, $M = 583$.

different resolutions. This figure shows the dynamics of a single structure that achieves the best fitness in one of the evolutionary trials in experiment 5 first as it was evolved, then regrown at a lower resolution and finally regrown once more at a higher resolution. It is noteworthy how in all three cases a mass distribution is preserved that allows the structure to first fall onto its heavier end, carry its momentum through a horizontal rotation and fall once again away from its starting position.

Unfortunately, not all of the evolved structures preserve their dynamics like this example when grown at different resolutions. Fig. 4.6 compares the performance of structures grown from CPPNs evolved in experiments 5 and 6 by re-querying at both a lower and higher resolution than that at which they were evolved. One can see how once again using the full set of CPPN inputs improves performance. When growing structures at a resolution lower than that used during evolution those grown from CPPNs using the

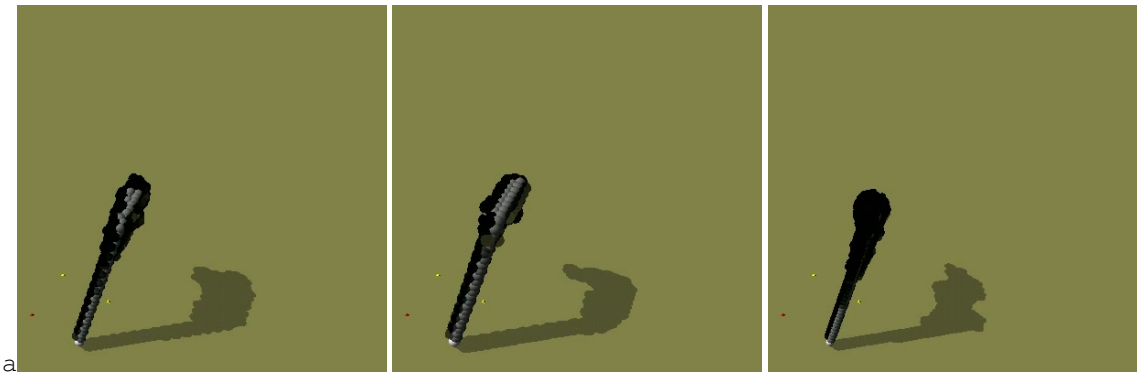


Figure 4.7: Enlarged snapshots of the structure shown in Fig. 4.5 grown at three different resolutions. Left to right: radius $r = 0.08$ meters, maximum $M = 391$ cells; $r = 0.1$m, $M = 200$; $r = 0.07$m, $M = 583$.

full set of inputs on average significantly outperform those grown from CPPNs which use the restricted set (experiment 5 vs. experiment 6). When growing structures at a higher resolution than what their CPPNs were evolved for no significant difference in performance is observed between those using the full set of inputs and those using the restricted set.

4.4 Conclusions

This paper demonstrates that CPPN-NEAT is capable of evolving three dimensional physical structures with non-trivial dynamical properties. Moreover a case has been made for why including additional inputs which recursively provide the CPPN with information about the growth trajectory and environment (those in the full set) is beneficial when using CPPN-NEAT to evolve such structures. Specifically, structures evolved using these inputs on average perform either equivalently or significantly better when compared with those grown from CPPNs that are limited to the basic Cartesian inputs in the restricted set. Moreover including these inputs results in CPPNs that are either as robust or more so to changes in growth resolution relative to CPPNs taking only the restricted set of inputs.

Additionally, this work demonstrates that it is possible to improve run time performance without significantly degrading the quality of evolved structures by using a lower resolution at the start of an evolutionary trial followed by increasing this resolution partway through. First evaluating structures at a lower resolution allows the evolutionary process to more quickly search the space of possible solutions before switching to a higher resolution to more further refine the shape and mass distribution of the structures.

This work is a first step in a research trajectory that aims to co-evolve articulated body plans and control policies. The results presented here are promising in this endeavor in that CPPN-NEAT is able to find non-intuitive solutions that capture the powerful yet subtle relationship between physical structure and function. Adding in articulation will involve extending the growth procedure presented here to additionally query evolved CPPNs for cell connectivity information. For example when adding a new cell to the morphology instead of always doing so in a rigid manner the CPPN can be queried with information regarding the two cells' geometric positions to produce another value used to determine whether the cells should be connected rigidly or with a rotational joint, and if they are to be connected with a joint, properties of this joint may also be determined from the CPPN output.

More work will be needed to extend the growth procedure to allow for the inclusion of arbitrary numbers of sensors on each cell, but the current results along with previous experiments using CPPNs suggest that additional CPPN outputs and/or input flags will provide the necessary mechanisms for extending the current framework in that direction. Additionally, since the HyperNEAT variant of CPPN-NEAT has shown success in the evolution of ANNs it is reasonable to expect that co-evolving neural network control policies along with the morphology should be possible. The authors intend to tackle these problems in future work.

4.5 References

Adamatzky, A., M. Komosinski, and S. Ulatowski (2000). Software review: Framsticks. *Kybernetes: The International Journal of Systems & Cybernetics 29*(9/10), 1344–1351.

Anderson, M. (2003). Embodied Cognition: A field guide. *Artificial Intelligence 149*(1), 91–130.

Beer, R. D. (1990). *Intelligence as adaptive behavior: an experiment in computational neuroethology*. San Diego, CA, USA: Academic Press Professional, Inc.

Beer, R. D. (2008). The dynamics of brain-body-environment systems: A status report. In P. Calvo and A. Gomila (Eds.), *Handbook of Cognitive Science: An Embodied Approach*, pp. 99–120. Elsevier.

Bongard, J. and R. Pfeifer (2001). Repeated structure and dissociation of genotypic and phenotypic complexity in Artificial Ontogeny. *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO)*, 829–836.

Bongard, J. C. (2002). Evolving modular genetic regulatory networks. In *Proceedings of The IEEE 2002 Congress on Evolutionary Computation (CEC2002)*, pp. 1872–1877.

Bongard, J. C. and R. Pfeifer (2003). Evolving complete agents using artificial ontogeny. In *Morpho-functional Machines: The New Species (Designing Embodied Intelligence*, pp. 237–258. Springer-Verlag.

Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. MIT Press.

Brooks, R. (1999). *Cambrian intelligence*. MIT Press Cambridge, Mass.

CGAFaq (2010). Evenly distributed points on sphere – cgafaq. [Online; http://cgafaq.info/wiki/Evenly_distributed_points_on_sphere accessed 25-January-2010].

Clune, J., B. Beckmann, C. Ofria, and R. Pennock (2009). Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computing*, pp. 2764–2771.

Clune, J., R. T. Pennock, and C. Ofria (2009). The sensitivity of HyperNEAT to different geometric representations of a problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

Collins, S., A. Ruina, R. Tedrake, and M. Wisse (2005). Efficient bipedal robots based on passive-dynamic walkers. *Science 307*(5712), 1082–1085.

Dellaert, F. and R. Beer (1994). Toward an evolvable model of development for autonomous agent synthesis. *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*.

Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. *Procs. of the Fourth European Conf. on Artificial Life*, 205–213.

Funes, P. and J. Pollack (1997). Computer evolution of buildable objects. *Fourth European Conference on Artificial Life*, 358–367.

Harvey, I., P. Husbands, D. Cliff, A. Thompson, and N. Jakobi (1997). Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems 20*, 205–224.

Hornby, G. S. and J. B. Pollack (2001a). Body-brain co-evolution using L-systems as a generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, pp. 868–875. Morgan Kaufmann.

Hornby, G. S. and J. B. Pollack (2001b). Evolving L-systems to generate virtual creatures. *Computers and Graphics 25*(6), 1041–1048.

Lipson, H. and J. Pollack (2000). Automatic design and manufacture of robotic lifeforms. *Nature 406*, 974–978.

Lund, H. H., J. Hallam, and W. Lee (1997). Evolving robot morphology. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*, Piscataway, NJ, USA. IEEE Press.

Mautner, C. and R. Belew (2000). Evolving robot morphology and control. *Artificial Life and Robotics 4*(3), 130–136.

Nelson, A. L., G. J. Barlow, and L. Doitsidis (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems 57*(4), 345–370.

Nolfi, S. and D. Floreano (2000). *Evolutionary Robotics: The Biology,Intelligence,and Technology*. Cambridge, MA, USA: MIT Press.

Pfeifer, R. and J. Bongard (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press.

Pollack, J. B., H. Lipson, G. Hornby, and P. Funes (2001). Three generations of automatically designed robots. *Artif. Life 7*(3), 215–223.

Saff, E. and A. Kuijlaars (1997). Distributing many points on a sphere. *The Mathematical Intelligencer 19*(1), 5–11.

Sims, K. (1994). Evolving 3D morphology and behavior by competition. *Artif. Life 1*(4), 353–372.

Stanley, K., D. D'Ambrosio, and J. Gauci (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life 15*(2), 185–212.

Stanley, K. and R. Miikkulainen (2003). A taxonomy for artificial embryogeny. *Artificial Life 9*(2), 93–130.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines 8*(2), 131–162.

Stanley, K. O. and R. Miikkulainen (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation 10*(2), 99–127.

Tedrake, R., T. Zhang, and H. Seung (2005). Learning to walk in 20 minutes. In *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, Yale University, New Haven, CT.

Watson, R. A., S. G. Ficici, and J. B. Pollack (1999). Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *Congress on Evolutionary Computation*, pp. 335–342. IEEE Press.

Chapter 5

Dynamic Resolution in the Co-Evolution of Morphology and Control

Evolutionary robotics is a promising approach to overcoming the limitations and biases of human designers in producing control strategies for autonomous robots. However, most work in evolutionary robotics remains solely concerned with optimizing control strategies for existing morphologies. By contrast, natural evolution, the only process that has produced intelligent agents to date, may modify both the control (brain) and morphology (body) of organisms. Therefore, co-evolving morphology with control may provide a better path towards realizing intelligent robots. This paper presents a novel method for co-evolving morphology and control using CPPN-NEAT. This method is capable of dynamically adjusting the resolution at which components of the robot are created. Advantages of this capability are demonstrated on a simple task, and implications for using this methodology to create more complex robots are discussed.

5.1 Introduction

There are many reasons why it would be useful to have autonomous robots operating in our homes and offices. These range from freeing people from repetitive tasks to the ability to perform actions that humans are incapable of. However, with the exception of a few robots designed to accomplish simple tasks, the

vast majority of autonomous robots currently in use operate only in factories and other highly structured environments. In order to make the migration out of the factories and into our everyday lives robots will need to be adaptive and exhibit intelligent behavior.

There has been much work in recent years in the area of embodied artificial intelligence (Brooks 1999, Anderson 2003, Pfeifer and Bongard 2006, Beer 2008) which has led to the conclusion that such intelligent behavior must arise out of the coupled dynamics between an agent's body, brain and environment. This means that the complexity of an agent's controller and morphology must increase commensurately with the task or tasks that it is required to perform. However, when designing complex autonomous robots it is often not clear how responsibility for different behaviors should be distributed across an agent's controller and morphology. A good example of this is that if a robot is solely tasked with moving over flat terrain while following a light source then wheels and a direct sensory motor mapping are an appropriate solution (Braitenberg 1986), but if the robot must be able to navigate over varied terrains while performing more complicated tasks a more complex control strategy and/or morphology are required. This issue of scaling up morphological and control complexity has been a major obstacle in developing autonomous robots capable of operating in most real world situations.

5.1.1 Background

The only truly intelligent agents to have yet existed, as far as we are aware, are biological organisms. Therefore the only known pathway to creating intelligent agents is evolution by natural selection. Guided by this observation, the field of evolutionary robotics (Harvey et al. 1997, Nolfi and Floreano 2000) attempts to realize intelligent agents by means of artificial evolution. Generally how this methodology works is that control policies for human designed or bio-mimicked robots are optimized to perform a desired task via evolutionary algorithms. This has allowed for the creation of robust, non-liner control strategies for autonomous agents that are not bound by the limits of human intuition. However, natural evolution does not operate on one part of an organism (brain) to the exclusion of others (body). In fact under evolution by natural selection any and all parts of an organism may be, and at some point in the past necessarily were, modified. This allows for the realization of organisms whose brains and bodies are co-optimized for specific ecological niches.

Luckily, artificial evolution is not necessarily limited to acting solely on a robot's brain or control strategy. Evolutionary frameworks in which the morphology and control of simulated machines are co-optimized in

virtual environments are possible and indeed have been created, starting with Sims (1994a) and followed by various other studies (Dellaert and Beer 1994, Lund et al. 1997, Adamatzky et al. 2000, Mautner and Belew 2000, Lipson and Pollack 2000, Hornby and Pollack 2001a, Hornby and Pollack 2001b, Stanley and Miikkulainen 2003, Eggenberger 1997, Bongard and Pfeifer 2001, Bongard 2002, Bongard and Pfeifer 2003). With this approach body plans and control policies uniquely suited for a machine's task environment may be found. This offers a substantial improvement over relying on body plans created by human designers who have inherent biases or copying animal body plans more suited to a given ecological niche.

The current work continues in this tradition while presenting several important advantages over previous approaches. First, the genomes of evolved agents are represented by compositional pattern producing networks (CPPNs) (Stanley 2007), a form of indirect encoding that have been shown able to capture geometric symmetries appropriate to the system being evolved, are capable of reproducing outputs at multiple resolutions (Stanley et al. 2009), and have shown promise in producing neural network control policies for legged robots (Clune et al. 2009, Clune et al. 2009). Second, through novel extensions of the CPPN outputs evolution can differentially optimize the resolution of the simulated robots such that a larger number of smaller sized components may be present in some body locations while a smaller number of larger sized components is present in other locations. To see why this is desirable consider evolving a creature capable of locomoting and grasping different objects. In this case evolution may choose to increase the resolution of the hands or grippers in order to achieve more fine grained control of the object to be grasped while at the same time using a lower resolution model of the trunk which will result in fewer components keeping the morphology from becoming unnecessarily complex and therefore providing faster simulations without sacrificing performance.

This paper extends the work presented in (Auerbach and Bongard 2010) to allow for evolution of control as well as dynamic resolution as just discussed. The paper is organized as follows: the next section describers the CPPN encodings used, describes how they are evolved and presents how these encoding are used to grow actuated robots. Following that a description of two experiments is presented which compare this dynamic resolution method with a similar method lacking this ability. Some observations of how evolution makes use of the dynamic resolution capability are discussed, and finally some conclusions and directions for future work are presented.

5.2 Methods

This section presents a brief description of CPPNs and the evolutionary algorithm used to evolve them. This is followed by a description of the methods used for generating actuated robots from evolved genotypes. After this a description is presented of the fitness function used for evaluating these robots.

5.2.1 CPPNs

Compositional Pattern Producing Networks (CPPNs) are a form of artificial neural network (ANN). Unlike most ANNs where each internal node uses a form of sigmoid function, each internal node of a CPPN can have an activation function drawn from a diverse set of functions. This function set includes functions that are repetitive such as sine or cosine as well as symmetric functions such as gaussian. By composing these functions CPPNs can produce motifs seen in the majority of natural systems such as symmetry, repetition, and repetition with variation. It is important to note that these motifs come out of this encoding for free without the need for a human expert to explicitly enforce or select for them.

5.2.2 CPPN-NEAT

In this work the CPPNs are evolved via CPPN-NEAT (Stanley 2007). CPPN-NEAT uses the NeuroEvolution of Augmenting Topologies (NEAT) method of neuro-evolution (Stanley and Miikkulainen 2002) to evolve increasingly complex CPPNs. An extension of CPPN-NEAT —HyperNEAT— has been used (Stanley et al. 2009, Clune et al. 2009, Clune et al. 2009) to evolve traditional ANNs, where each node of the ANN is embedded in a geometric space and whose coordinates are fed to an evolved CPPN to determine the presence and weights of connections. In effect these connections are “painted” on to the network from the output patterns produced by the CPPN. As shown by Stanley et al. (2009) this has the crucial benefit that a CPPN evolved to produce the connectivity patterns of small ANNs can be re-queried at a higher resolution to produce the connectivity patterns of larger ANNs without needing to re-evolve these large ANNs. Similarly as shown in (Auerbach and Bongard 2010) it is possible to change the resolution at which CPPNs grow physical structures.

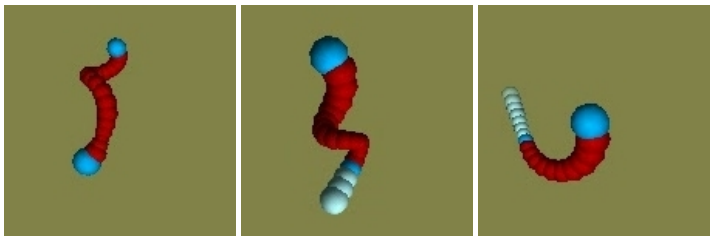


Figure 5.1: A few samples of robots evolved for directed locomotion.

5.2.3 Growing Actuated Robots from CPPNs

In this work actuated robot morphologies and control strategies are grown from evolved CPPNs. Each robot is composed of many spherical cells which connect to each other either rigidly or via single degree of freedom rotational joints. For an example of robots produced in this way see Figure 5.1.

The growth procedure begins with a single cell, henceforth referred to as the root, with a predefined radius r_{init} located at a designated origin. A cloud composed of n points is cast around this cell with the n points being evenly distributed on the surface of the root sphere (all n points are at distance r from the center of the root). In the current work, n is restricted to 2, such that the points are directly opposite each other along the y-axis. In the coordinate system used here z is the vertical axis, and so the y-axis represents a horizontal axis that passes through the center of each cell. It is convenient to think of this as a cloud of points though, as is the case in (Auerbach and Bongard 2010), because in future work this restriction will once again be lifted allowing for a greater number of morphologies.

Once this cloud is cast, every point in the cloud is used to query a CPPN. The CPPN is queried by providing as input the Cartesian coordinates (x, y, z) of the point in question, the radius r_{parent} of the sphere to which it will attach ($r_{\text{parent}} = r_{\text{init}}$ when considering points around the root), and a constant bias input. These values are propagated through the CPPN to produce multiple output values. The first of these outputs is m. This output value can be thought of as a concentration of matter at that point, such that when m is over a certain matter threshold, T_{matter}, a cell will be placed at that point. The more that m exceeds the matter threshold the denser the cell placed at that point will be. This creates a continuum from no cell existing at that location up to having a very dense cell at that location with all intermediate levels of density in between being possible. The second of these outputs is a radius scaling factor r_{scale} which will determine the size of the cell to be added at that location.

```
1.  GrowRobot(CPPN)
2.      Initialize priority queue q, with priority based on cell density
3.      Create cell c at origin with full density and radius r_init, add to morphology M,
            and flag its coordinates 'discovered'
4.      Enqueue c in q
5.      WHILE ~ q.isEmpty
6.          c ← q.front
7.          Cast point cloud C centered at c
8.          Initialize vector V of neighboring cells
9.          FOR EACH point p in C
10.             Query CPPN at p to get output values m and r_scale
11.             Add p with values m and r_scale to vector V
12.         Sort V by descending value of m
13.         FOR EACH point p with value m in sorted vector V
14.             IF coordinates of p not yet 'discovered'
15.                 Flag p 'discovered'
16.                 IF CanAdd(p,m,c,r)
17.                     Add cell centered at p with density ∝ m and radius r = r_parent * r_scale to morphology M
18.                     Re-query CPPN at (c+p)/2 to get output values j, θ and Δ.
19.                     IF j > T_joint
20.                         Determine joint normal n⃗ from θ
21.                         Connect cell with 1-DOF rotational joint with normal n⃗, range ∝ j actuated by
                                CPG with phase offset ∝ Δ
22.                     ELSE
23.                         Connect cell rigidly
24.                     Enqueue (p, v) in q
25. CanAdd(p, m, c, r)
26.     IF m > T_matter AND ∀ cells d ∈ M, d ≠ c dist(p,d) ≥ r AND p is within bounding cube
27.       Return true
28.     ELSE
29.       Return false
```

Figure 5.2: **Grow Robot pseudo code.** The growth procedure starts with a root cell at the origin (line 3). Then, as long as there are cells in the queue to consider it takes the cell at the front of the queue, casts a point cloud around it and considers adding a cell at each point in turn (lines 5-17). A cell is added at a given point if all of the following hold: it does not conflict with a previously added cell, the CPPN outputs a value above the threshold T_{matter} when queried at that point, and the point is within the bounding cube (lines 25-29). If a cell is to be added the CPPN is queried once again to determine connectivity and control parameters (lines 18-23).

Once the m and r_{scale} values have been determined for all n points in the cloud the points are sorted in descending order of the matter output m. The sorted points are then looped through and the algorithm considers adding a cell centered at each point in turn. Specifically a cell, centered at point p is added to the structure if (a) the output value of point p is above the threshold T_{matter} and (b) no other cell, besides the one to which this new cell will be attached (its parent) has previously been added to the structure with center located at distance $< r$ away from p.

The radius r of a cell is determined from the radius of its parent r_{parent} and the output value r_{scale}. Specifically

$$r = \begin{cases} r_{\text{parent}} * r_{\text{scale}} & r_{\text{min}} \leq r_{\text{parent}} * r_{\text{scale}} \leq r_{\text{max}} \\ r_{\text{min}} & r_{\text{parent}} * r_{\text{scale}} < r_{\text{min}} \\ r_{\text{max}} & r_{\text{parent}} * r_{\text{scale}} > r_{\text{max}} \end{cases} \tag{5.1}$$

That is, the cell to be added will have radius equal to that of its parent scaled by a factor determined by the CPPN output capped by a minimum and maximum possible radius.

If a cell has been selected for addition to the robot the CPPN will be queried once more to determine connectivity and control parameters. In particular the CPPN will be fed the coordinates where a joint may be added: a cell centered at point p connecting to a parent cell centered at point p_{parent} may be connected by a single degree of freedom (DOF) rotational joint located halfway between p and p_{parent} ($\frac{p+_{\text{parent}}}{2}$). These coordinates are input to the CPPN along with r_{parent} to retrieve additional outputs: a joint "concentration" j, an angle θ and a phase offset Δ.

If the output j exceeds a joint threshold T_{joint} the cell will attach to its parent with a 1-DOF rotational joint. The more j exceeds this threshold the greater the range of motion of the connecting joint will be. Similar to the matter case this creates a continuum from connecting rigidly when $j \leq T_{\text{joint}}$ to connecting via a joint with a very narrow range to connecting via a joint with a large range of motion.

If indeed a given cell will connect to its parent via a joint there are two more important properties of this connection to be determined. First, the direction of motion of this joint is defined by a normal vector $\vec{n}$. This vector will be normal to the axis $\vec{a}$ defined by the center of the cell and the center of its parent. To choose one vector out of the infinitely many such vectors the cross product of $\vec{a}$ and a default vector $\vec{d}$ is taken. This results in a single vector normal to $\vec{a}$ which is then rotated around $\vec{a}$ by angle θ. In this way all possible vectors normal to $\vec{a}$ may be used in constructing the joint and it is left up to the CPPN to output a single angle to choose a specific normal vector.

The second property to be determined in the case where a cell connects via a joint is what control signal drives the motor actuating this joint. In this work all motors are controlled by time dependent harmonic oscillators. A central sinusoidal oscillation is used, but each individual motor is allowed to be out of phase

with this central control signal. The phase offset of each motor is determined by the final CPPN output Δ when queried at the joint's location. In this way the CPPN also determines the control policy of the robot being grown in addition to its morphology.

Once a cell is added to the structure and its connectivity and control have been determined it gets placed into a priority queue whose priority is based on its matter concentration m. When all points from the current cloud have been considered the algorithm takes the cell at the top of the priority queue and casts a point cloud around it, and this process continues until there are no valid possible points at which to place cells. Points are valid if they are within a bounding cube with side lengths l. This bounding cube constraint was imposed so that in the future it will be possible to physically fabricate the entire evolved robots within the confines of a 3D-printer. Figure 5.2 gives pseudo code for this growth procedure.

There are several reasons why it is desirable to have a growth procedure such as this. Merely querying CPPNs over a sampling of three-dimensional space may lead to disconnected objects. Even if all but one of these objects are thrown out much computational resources will have been wasted querying these regions of space. Additionally, imposing a grid over space to determine which points to query imposes a specific resolution on the morphology and thus removes much of the benefit of the dynamic resolution (radius) method used in this work because the spacing of the cells will have been predetermined by the grid.

5.2.4 Selecting for robots with desirable properties

This paper aims to demonstrate that CPPN-NEAT coupled with the growth procedure just presented is capable of evolving actuated robot morphologies and control policies for a given task. In particular the property selected for in this work is maximum directed displacement of the robot in a fixed amount of time.

To select for this property, an evolved virtual robot is placed in a physical simulator[1] for that set amount of time. The fitness of this robot (and hence its encoding CPPN) that CPPN-NEAT attempts to maximize is simply the y-coordinate of the robot's center of mass after the simulation completes subject to a few conditions. The first of these conditions is to prevent robots from exploiting simulation faults. There are a number of ways these faults could be avoided such as reducing the step size used in running the simulation, but this would lead to increased simulation runtimes. The technique used here is to throw out any solution where the robot's linear or angular acceleration exceed predefined thresholds by giving 0 fitness. The second

[1] Simulations are conducted in the Open Dynamics Engine (`http://www.ode.org`), a widely used open source, physically realistic, simulation environment

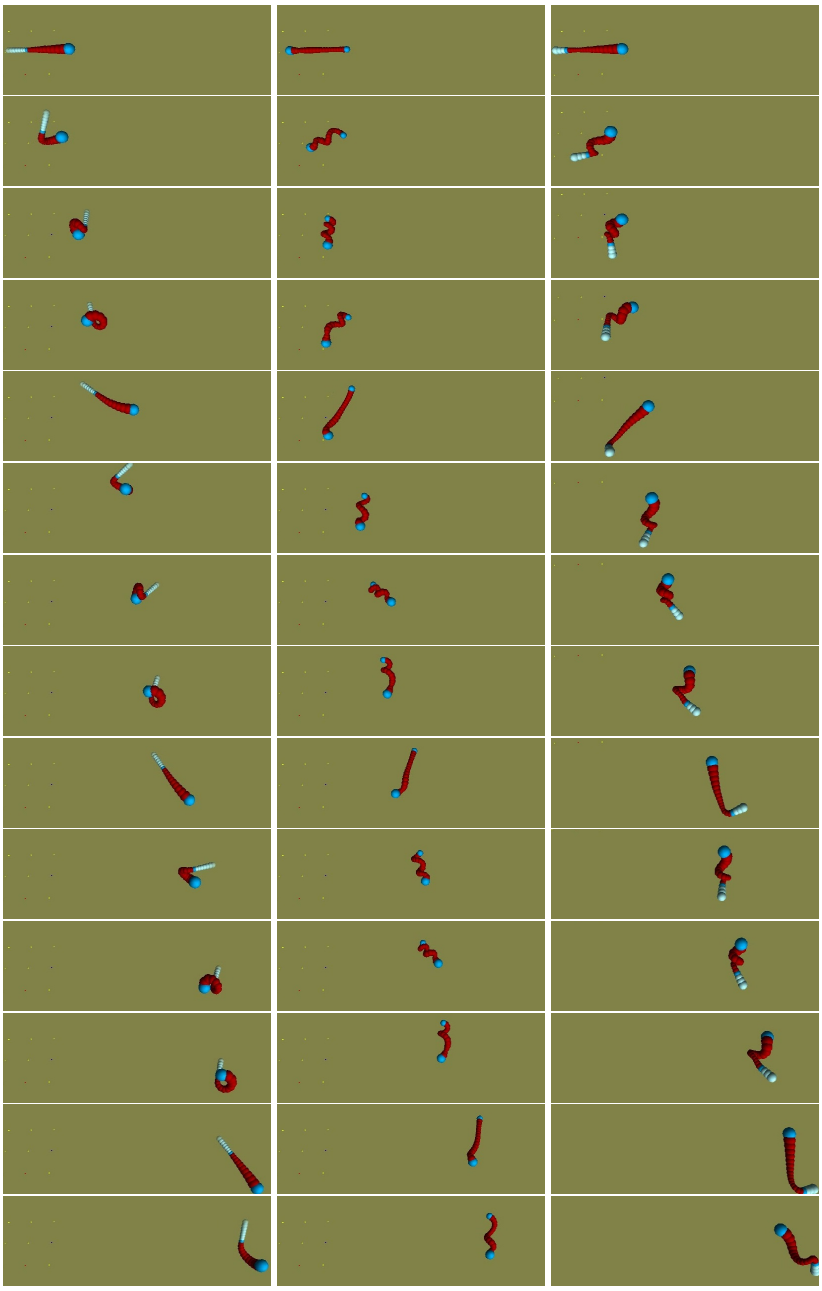


Figure 5.3: Behavior of a few of the more successful robots. Each column shows the behavior of a different dynamic resolution robot evolved for directed locomotion (with time going from top to bottom). Three different robots are shown. Red cells are attached to two joints while the darker blue cells attach to a single joint. The lighter blue cells all connect rigidly. Enlarged pictures of each of these robots are shown in Fig. 5.1.

condition is to prevent solutions where the robot moves by rolling on a subset of its cells. These solutions tend to be common but are less interesting than other solutions that may be found, therefore any robot that has a subset of its cells remain in contact with the ground for over 95% of the time is discarded and given a fitness of 0 once again.

5.3 Results

This section presents experiments comparing how the dynamic resolution method presented above performs in comparison to a similar method restricted to using cells with a fixed radius. It should be noted that using a fixed radius in this case would be equivalent to omitting the growth procedure and merely querying the evolved CPPN over a gridded region of space and then taking those cells which connect to the cell at the origin as the resulting morphology, however as mentioned above this procedure would require more computational resources than using the growth procedure to accomplish the same result.

Specifically, two experiments are conducted each consisting of a set of 30 evolutionary trials. All experiments attempt to evolve simulated robots with CPPN-NEAT capable of directed locomotion using the fitness criteria presented above. Moreover, all experiments are configured to use a population size of 150, and run for 500 generations with each fitness evaluation given 2500 time steps. Additionally in all experiments the values T_{matter} and T_{joint} are both fixed at 0.7, and each cell of the structure is restricted to having its center initially located in interval $(0, [-2, 2], 0)$ (coordinates all in meters). Before being placed in the simulator the morphologies are translated vertically such that the largest component is resting on the ground. The CPPN internal nodes are allowed to use the signed cosine, gaussian, and sigmoid activation functions. All other parameters of the evolutionary algorithm are kept at the default values provided with the C++ implementation of HyperNEAT[2].

The trials in the first experiment grow structures using the dynamic resolution method introduced in this paper. In this case r_{init} was set to 0.1 meters, r_{min} set to 0.01 meters, and r_{max} set to 0.5 meters. Additionally the output value r_{scale} is normalized to the range $[0.5, 1.5]$; that is, a newly added cell can have radius at the most 50% larger and at the least 50% smaller than its parent. Figure 5.3 demonstrates the behavior of a few of the more successful robots to evolve in evolutionary trials in this experiment.

[2]Available at `http://eplex.cs.ucf.edu/hyperNEATpage/HyperNEAT.html`

The second experiment is exactly the same as the first one, but it is restricted to growing robots composed of cells with a fixed radius. CPPN-NEAT is still used to evolve CPPNs which are used to grow the morphologies and control strategies under the procedure outlined above, but the r_{scale} output is not included in the CPPNs. In lieu of determining cell size from this output this experiment builds robots from cells all having radius $r_{\text{fixed}} = 0.1$ meters.

5.4 Discussion

One advantage of using the dynamic resolution method over keeping resolution fixed is that it allows evolution to explore a greater variety of possible solutions. The first evidence of this is observational. Looking at the behavior of the three robots shown in Figure 5.3 a variety of dynamics can be observed. The leftmost robot resembles a whip in that it has one thicker end and tapers off to a thinner end. Additionally we see that the thin end is rigid. This can be inferred from the light blue coloring of the cells at that end which represent cells that are not connected to any joint (while red cells connect to two joints and dark blue cells to a single joint). Scanning down the panels one can see that this rigid end is utilized as a paddle to propel the robot forward while curling over at the other end.

The middle robot on the other hand has no rigid connections. This robot moves by coiling and uncoiling to move itself in the desired direction. The rightmost robot has yet a different morphology and movement pattern than the other two. While it has one rigid end like the leftmost robot this end is composed of fewer spheres and actually includes cells that are larger than those in the middle of its body, flaring back out like a baseball bat. This configuration is actually the most successful one discovered and its movement pattern is different from the other two robots.

Additional evidence of the dynamic resolution runs exploring a greater variety of morphologies is shown in Figure 5.4. The top part of this figure shows the mean number of cells used by the best individual from each generation across the 30 evolutionary trials from both the dynamic resolution set and the fixed resolution set. The bottom portion of this figure shows the standard deviation from the means shown in the top. One can see here that the trials in the dynamic resolution set tend to explore morphologies with a large number of small cells early on, followed by exploring a fewer number of cells on average later on in the trials. However, while the fixed resolution robots tend to converge to a narrow range of cell numbers as exemplified by the

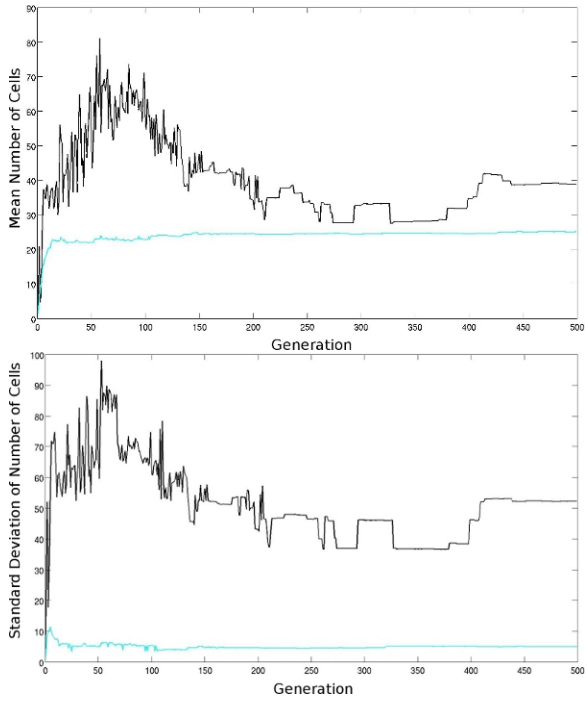


Figure 5.4: Dynamic resolution runs explore a greater variety of morphologies. **Top**: Mean number of cells of best individual in each generation across the 30 evolutionary trials for the dynamic resolution set (black) and the fixed resolution set (light blue). **Bottom**: Standard deviation from the mean number of cells by generation.

constant mean and small standard deviation, the dynamic resolution robots continue to explore a wide array of different number of cells and cell sizes which can be inferred by observing that their standard deviation never comes back down.

This evidence is corroborated by Figure 5.5 which plots the mean and standard deviation of cell radii *within* each best of generation individual averaged across the 30 evolutionary trials. Here it is shown in a different way how the dynamic runs tend to explore smaller cell sizes early on in the evolutionary trials followed by larger cell sizes later. While this is the case on average, by looking at the standard deviations we see that as evolution progresses morphologies with a wide variety of cell sizes come into being (the standard deviation trends upwards). This means that the dynamic resolution runs are exploring the space of solutions with variable cell sizes which is not possible in the fixed resolution case.

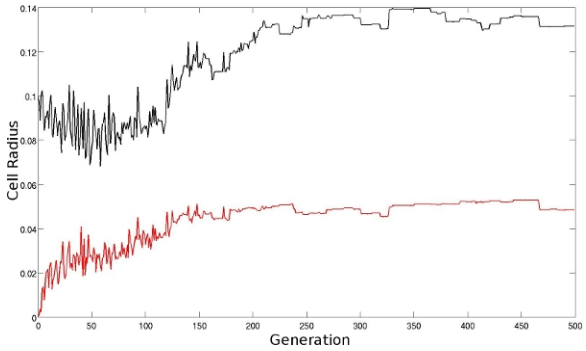


Figure 5.5: Changes in cell radii over evolutionary time. Mean (black) and standard deviation from the mean (red) of cell radii *within* each best of generation individual from the dynamic resolution set averaged across the 30 evolutionary trials.

5.5 Conclusion

This paper has demonstrated how one can implement a growth mechanism that can generate robots composed of variable sized components. This ability was then shown to be actually utilized by demonstrating how evolutionary trials that incorporate this dynamic resolution mechanism explore a greater variety of possible solutions than evolutionary trials that are restricted to constructing robots out of fixed sized components.

While it is not directly evident what performance advantage using dynamic resolution offers on a task as simple as the one utilized in this work, intuitively one can see the benefit of such a mechanism when generating more complex robots for more complex tasks. Specifically in any task that requires object manipulation it will be useful to adapt the component sizes of the parts of the morphology that will be in contact with external objects while not creating overly complex morphologies as would be the case if such a high resolution were employed for the entire robot. Additionally, it may not be possible to know the ideal component size *a priori*, and so using a dynamic resolution method such as this can help steer evolution towards constructing robot morphologies with the proper component sizes.

Much work remains to be done in exploring the possibilities of this methodology. The logical next step will be to relax some of the restrictions imposed in this work such as allowing robots to grow in arbitrary trajectories as opposed to along only a single axis. The authors additionally plan to tackle more complex tasks including object manipulation to test whether using dynamic resolution will result in the additional predicted advantages discussed here. This will require the use of more complex control strategies such as

neural networks, and the inclusion of a mechanism for endowing the robots with sensors in order to close the control loop. The methods used here for generating joint and motor parameters via additional CPPN outputs seem promising and the authors plan to further leverage this technique for determining sensor and neuron positions and parameters.

5.6 References

Adamatzky, A., M. Komosinski, and S. Ulatowski (2000). Software review: Framsticks. *Kybernetes: The International Journal of Systems & Cybernetics 29*(9/10), 1344–1351.

Anderson, M. (2003). Embodied Cognition: A field guide. *Artificial Intelligence 149*(1), 91–130.

Auerbach, J. E. and J. C. Bongard (2010). Evolving CPPNs to grow three-dimensional physical structures. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Beer, R. D. (2008). The dynamics of brain-body-environment systems: A status report. In P. Calvo and A. Gomila (Eds.), *Handbook of Cognitive Science: An Embodied Approach*, pp. 99–120. Elsevier.

Bongard, J. and R. Pfeifer (2001). Repeated structure and dissociation of genotypic and phenotypic complexity in Artificial Ontogeny. *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO)*, 829–836.

Bongard, J. C. (2002). Evolving modular genetic regulatory networks. In *Proceedings of The IEEE 2002 Congress on Evolutionary Computation (CEC2002)*, pp. 1872–1877.

Bongard, J. C. and R. Pfeifer (2003). Evolving complete agents using artificial ontogeny. In *Morpho-functional Machines: The New Species (Designing Embodied Intelligence*, pp. 237–258. Springer-Verlag.

Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. MIT Press.

Brooks, R. (1999). *Cambrian intelligence*. MIT Press Cambridge, Mass.

Clune, J., B. Beckmann, C. Ofria, and R. Pennock (2009). Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computing*, pp. 2764–2771.

Clune, J., R. T. Pennock, and C. Ofria (2009). The sensitivity of HyperNEAT to different geometric representations of a problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

Dellaert, F. and R. Beer (1994). Toward an evolvable model of development for autonomous agent synthesis. *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*.

Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. *Procs. of the Fourth European Conf. on Artificial Life*, 205–213.

Harvey, I., P. Husbands, D. Cliff, A. Thompson, and N. Jakobi (1997). Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems 20*, 205–224.

Hornby, G. S. and J. B. Pollack (2001a). Body-brain co-evolution using L-systems as a generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, pp. 868–875. Morgan Kaufmann.

Hornby, G. S. and J. B. Pollack (2001b). Evolving L-systems to generate virtual creatures. *Computers and Graphics 25*(6), 1041–1048.

Lipson, H. and J. Pollack (2000). Automatic design and manufacture of robotic lifeforms. *Nature 406*, 974–978.

Lund, H. H., J. Hallam, and W. Lee (1997). Evolving robot morphology. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*, Piscataway, NJ, USA. IEEE Press.

Mautner, C. and R. Belew (2000). Evolving robot morphology and control. *Artificial Life and Robotics 4*(3), 130–136.

Nolfi, S. and D. Floreano (2000). *Evolutionary Robotics: The Biology,Intelligence,and Technology*. Cambridge, MA, USA: MIT Press.

Pfeifer, R. and J. Bongard (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press.

Sims, K. (1994). Evolving 3D morphology and behavior by competition. *Artif. Life 1*(4), 353–372.

Stanley, K., D. D'Ambrosio, and J. Gauci (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life 15*(2), 185–212.

Stanley, K. and R. Miikkulainen (2003). A taxonomy for artificial embryogeny. *Artificial Life 9*(2), 93–130.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines 8*(2), 131–162.

Stanley, K. O. and R. Miikkulainen (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation 10*(2), 99–127.

Chapter 6

Evolving Complete Robots with CPPN-NEAT: The Utility of Recurrent Connections

This paper extends prior work using Compositional Pattern Producing Networks (CPPNs) as a generative encoding for the purpose of simultaneously evolving robot morphology and control. A method is presented for translating CPPNs into complete robots including their physical topologies, sensor placements, and embedded, closed-loop, neural network control policies. It is shown that this method can evolve robots for a given task. Additionally it is demonstrated how the performance of evolved robots can be significantly improved by allowing recurrent connections within the underlying CPPNs. The resulting robots are analyzed in the hopes of answering why these recurrent connections prove to be so beneficial in this domain. Several hypotheses are discussed, some of which are refuted from the available data while others will require further examination.

6.1 Introduction

6.1.1 Motivation

If robots could operate autonomously in outdoor or other unstructured environments such as the home or office they would be of great social utility. However, the vast majority of robots currently in operation are confined to performing pre-programmed actions in structured environments such as factories.

The principles of embodied artificial intelligence dictate that intelligent behavior must arise out of the coupled dynamics between an agent's body, brain and environment (Brooks 1999, Anderson 2003, Pfeifer and Bongard 2006, Beer 2008). This means that the complexity of an agent's control policy and physical body (morphology) must be proportional to the tasks it needs to perform. This poses a challenge when dealing with complex agents acting in complex environments: it is not always clear how responsibility for different behaviors should be distributed across the agent's controller and morphology.

6.1.2 Background

By applying evolutionary algorithms to optimize robot control policies, evolutionary robotics (Harvey et al. 1997, Nolfi and Floreano 2000) provides a framework for overcoming the limitations of human intuition in designing robust, non-linear, control strategies. While most evolutionary robotics projects have restricted themselves to optimizing control strategies for human designed or bio-mimicked morphologies, evolutionary algorithms may also be used to design complete robots including their physical morphologies in addition to their control policies.

Sims (Sims 1994) was the first to introduce an evolutionary framework in which both the morphology and control of simulated machines could be evolved in virtual environments to produce adaptive behavior. This work has been followed by other studies (Dellaert and Beer 1994, Lund et al. 1997, Adamatzky et al. 2000, Mautner and Belew 2000, Lipson and Pollack 2000, Hornby and Pollack 2001a, Komosinski and Rotaru-Varga 2002, Hornby and Pollack 2001b, Stanley and Miikkulainen 2003, Eggenberger 1997, Bongard and Pfeifer 2001, Bongard 2002, Bongard and Pfeifer 2003, Auerbach and Bongard 2010a) which also explored evolving both the morphology and control policy of robots in virtual environments. This approach of evolving both morphology and control has the advantage of being able to discover body plans uniquely suited for the

machine's task environment rather than being artifacts of human design biases or reproductions of biological morphologies that are only appropriate for that animal's ecological niche. And, importantly, this approach is not restricted to creating virtual robots, but can be applied to creating real, physical, robots through the use of rapid prototyping technologies as demonstrated by Lipson and Pollack (Lipson and Pollack 2000).

Like these previous studies, the current work also aims to evolve complete robot morphologies and controllers in virtual environments. While the approach presented here takes inspiration from these other studies, the methods employed are distinct in important ways which offer advantages over previous approaches. The most important distinction is the type of genomic encoding utilized.

Many of the studies in evolving morphologies and controllers have used indirect or generative genetic encodings. These have included models of genetic regulatory networks (Eggenberger 1997, Bongard and Pfeifer 2001, Bongard 2002, Bongard and Pfeifer 2003), meta-graphs (Sims 1994), and context-free grammars (Hornby and Pollack 2001a). Specifically it has been demonstrated (Hornby and Pollack 2001b, Komosinski and Rotaru-Varga 2002) that such encodings offer demonstrable benefits in this domain over direct encodings. Among other advantages, indirect encodings can more compactly represent complex structures and can provide pathways to creating reusable components.

In the work presented here the genomes of evolved agents are compositional pattern producing networks (CPPNs) (Stanley 2007). CPPNs are a form of indirect encoding that have several desirable properties. They have been shown able to capture geometric symmetries appropriate to the system being evolved, are capable of reproducing outputs at multiple resolutions (Stanley et al. 2009, Auerbach and Bongard 2010b), and have shown promise in producing neural network control policies for legged robots (Clune et al. 2009, Clune et al. 2009). The combination of these features makes it likely that evolving CPPNs will prove to be a more promising approach to realizing intelligent agents than other previous approaches.

Further advantages can be gained by extending CPPNs so that evolution can differentially optimize the resolution of the simulated robots, as demonstrated in (Auerbach and Bongard 2010a). In this case resolution refers to the size and quantity of a robot's components. This allows a large number of small sized components to be present in some body locations while a smaller number of larger sized components is present in other locations. As an example of why this is desirable, consider evolving a creature capable of locomoting and grasping different objects. In order to achieve a high degree of control of the object to be grasped the robot will need to have highly resolved hands or grippers, however the main body of the robot may not require

such high resolution. Without the ability to use different resolutions the entire morphology would need to be as highly resolved as these grippers, which would lead to an overly large degree of complexity in the morphology, which would in turn slow down the simulation. With this ability, on the other hand, a lower resolution model of the main body can be used which will result in fewer components thus keeping the morphology from becoming unnecessarily complex, and therefore providing for faster simulations without sacrificing performance.

This paper extends the work of (Auerbach and Bongard 2010a) by allowing CPPNs to encode embedded neural network controllers as well as a variety of sensor modalities. Additionally, in the current work the space of possible morphologies is extended from the simple single axis morphologies reported in (Auerbach and Bongard 2010a), and an additional class of encoding CPPNs is investigated.

The paper is organized as follows: the next section further describes the CPPN encodings used, describes how they evolve and how they produce actuated robots with embedded neural network controllers. Following that results are presented which compare using basic feed-forward CPPNs to those that allow for recurrent connections. It is shown that recurrent connections are useful in this domain, which brings up the question of why this is so. The next section discusses potential answers to this question through an analyses of the evolved robots. Finally a conclusion is presented and directions for future work are discussed.

6.2 Methods

6.2.1 CPPNs

Compositional Pattern Producing Networks (CPPNs) (Stanley 2007) are a form of artificial neural network (ANN) where each internal node can have an activation function drawn from a diverse set of functions as opposed to being limited to a standard sigmoid as is the case with classical ANNs. This function set includes functions that are repetitive such as sine or cosine as well as symmetric functions such as Gaussian, thus allowing for motifs seen in natural systems: symmetry, repetition, and repetition with variation. A thorough description of CPPNs is beyond the scope of this paper, the reader is referred to (Stanley 2007) for a more detailed explanation.

6.2.2 Evolutionary Algorithm

In this work CPPNs are evolved using CPPN-NEAT: an extension of the NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen 2002), method of neuro-evolution. Though a description of NEAT is beyond the scope of this paper (the reader is referred to (Stanley and Miikkulainen 2002) for a complete description) it is important to note that CPPN-NEAT begins with small CPPNs (those without any internal or hidden nodes) and gradually increases the complexity of the CPPNs over time through the addition of new nodes and links while dividing the population into "species" for the purpose of promoting genotypic diversity and allowing novel structural innovations time to mature.

6.2.3 Growing Robot Morphologies and Controllers from CPPNs

In this work actuated robot morphologies and neural network control strategies are grown from the evolved CPPNs. Each robot is composed of many spherical components with embedded sensors and neurons (see Figure 6.4 at the end of the paper for some example morphologies). The components connect to each other either rigidly or via actuated single degree of freedom rotational joints. The robots are controlled in a closed-loop fashion via embedded neural networks. Specifically the embedded neural networks are Continuous Time Recurrent Neural Networks (CTRNNs) (Beer 2006).

CTRNNs are a form of ANN where each neuron has an internal time constant, τ, and whose updates are governed by a set of differential equations as opposed to updating at discrete time steps. Additionally CTRNNs contain recurrent connections, and thus are capable of a form of memory, in contrast to traditional feed-forward ANNs where such connections are not allowed.

The growth procedure begins with a single component, henceforth referred to as the root, with a predefined radius r_{init} located at a designated origin. A cloud composed of n equally spaced points is cast around the root such that the n points are located on the surface of the root sphere (all n points are at distance r_{init} from the center of the root). In the current work, the n points are all located on the horizontal (x-y) plane at an interval of 0.01 radians for a total of $n = 629$ points.

Once this cloud is cast, every point in the cloud is used to query the CPPN. The CPPN is queried by providing as input the Cartesian coordinates (x, y, z) of the point in question, the radius r_{parent} of the sphere to which it will attach ($r_{\text{parent}} = r_{\text{init}}$ when considering points around the root), and a constant bias input.

Table 6.1: Description of sensor types. Each sphere may contain any subset of these four sensor types.

Sensor Type	Sensor Function
Distance	Senses the distance to a target: the target emits a sound, and when the target is within sensor range the distance sensor will output a value proportional to the sound's volume.
Touch	Binary sensor that outputs one when the sphere containing this sensor is touching the ground, an external object or any other body part it is not immediately attached to. Otherwise outputs 0.
Proprioceptive	This sensor is restricted to being placed in spheres that connect to their parent via a joint. Outputs a value proportional to the current angle of that joint.
Time	Outputs a sinusoidal oscillation over time.

These values are propagated through the CPPN to produce multiple output values. The first of these outputs is used to determine the "concentration" of matter at this point and is denoted m. When m is over a certain matter threshold, T_{matter}, it is possible that a sphere will be placed at that point. The more that m exceeds the matter threshold the denser a sphere at that point will be. This creates a continuum from no matter existing at that location up to having a very dense sphere at that location with the possibility of having any intermediate level of density in between. The second of these outputs is a radius scaling factor r_{scale} which will determine the radius, r, of the sphere to be added at that location and therefore provides for differential resolution as discussed above.

Once the m and r_{scale} values have been determined for all n points in the cloud the points are sorted in descending order of the matter output m. The sorted points are then looped over as the algorithm considers adding a sphere centered at each point in turn. Specifically a sphere, centered at point p, is added to the morphology if (a) the output value m of point p is above the threshold T_{matter} (b) no other sphere, besides the one to which this new sphere will be attached (its parent) has previously been added to the morphology with center located at distance $< r$ away from p, (c) no sphere belonging to a different rigid component (with the exception of those directly connected by a joint) will interpenetrate this new sphere and (d) this new sphere remains within a predefined bounded area.

The radius r of a sphere is determined from the radius of its parent r_{parent} and the output value r_{scale}. Specifically

$$r = \begin{cases} r_{\text{parent}} * r_{\text{scale}} & r_{\text{min}} \leq r_{\text{parent}} * r_{\text{scale}} \leq r_{\text{max}} \\ r_{\text{min}} & r_{\text{parent}} * r_{\text{scale}} < r_{\text{min}} \\ r_{\text{max}} & r_{\text{parent}} * r_{\text{scale}} > r_{\text{max}} \end{cases} \tag{6.1}$$

That is, the sphere to be added will have radius equal to that of its parent scaled by a factor determined by the CPPN output capped by a minimum and maximum possible radius. This procedure is employed to allow for dynamic resolution without the possibility of drastically different sized spheres being connected to each other. If the CPPN were to output the radius of each sphere directly then a sphere might completely engulf its neighbor which would create additional challenges for the physical simulation such as creating invalid interpenetrations.

Once a sphere has been selected for addition to the robot a third CPPN output, j, which dictates the presence of a joint is considered. If the output j exceeds a joint threshold T_{joint} the sphere will attach to its parent with a 1-DOF rotational joint located at the child sphere's center, otherwise it will be fused to its parent. The more j exceeds the threshold the greater the range of motion of the connecting joint will be. Similar to the matter output this creates a continuum from connecting rigidly when $j \leq T_{\text{joint}}$ to connecting via a joint with a very narrow range to connecting via a joint with a large range of motion.

If indeed a given sphere will connect to its parent via a joint there are several more CPPN outputs which are considered to determine how this joint is created and how it is controlled. First, the direction of motion of this joint is determined by an output θ. θ is used to determine a vector $\vec{n}$ that is normal to the joint's direction of motion. Since it is desirable that this vector be normal to the axis $\vec{a}$ defined by the center of the sphere and the center of its parent the cross product of $\vec{a}$ and a default vector $\vec{d}$ is taken. This results in a single vector normal to $\vec{a}$ which is then rotated around $\vec{a}$ by angle θ. In this way all possible vectors normal to $\vec{a}$ may be used in constructing the joint and it is left up to the CPPN to output a single angle θ which determines a specific normal vector.

When a joint is created a corresponding motor neuron which will control the joint's actuation is also created. In this case two additional CPPN outputs are considered to determine properties of this motor

neuron. These outputs are τ and ω. τ defines the time constant, and ω the bias of this motor neuron in the underlying CTRNN. The connectivity of this motor neuron to the remainder of the CTRNN controller is determined after the entire morphology is created, and is described below.

Whether or not a sphere is connected with a joint or not, it has the possibility of having one or more sensors embedded in it. Specifically, in the current work, four types of sensors are allowed and the presence or absence of each type of sensor within the current sphere is determined by a dedicated CPPN output. These sensor types are described in Table 6.1. Once again if a sensor is added its connectivity to the controller is determined after the entire morphology is created (see below).

Once a sphere is added to the morphology and its connectivity, sensor and possible neural network parameters have been determined it gets placed into a priority queue whose priority is based on its matter concentration m. When all points from the current cloud have been considered the algorithm takes the sphere at the top of the priority queue and casts a point cloud around it and the above procedure repeats. This process continues until there are no possible points at which to place spheres or until a maximum number of spheres (M) has been reached.

Once all spheres have been added through this process there are a few additional steps taken to complete the construction of the robot. The first of these steps involves pruning joints which connect "leaf" spheres[1] to their parents. This is done to prevent the creation of morphologies with "invalid" behaviors. Consider such a sphere, a, attached to its parent via a joint. Recall from above that the fulcrum of a joint connecting a sphere to its parent is located at the center of the child sphere. This means that a, not being connected to any other spheres, will simply spin in place when the joint connecting it to its parent is actuated. The types of behaviors achievable with such joints are undesirable and so such joints are removed, their motor neurons are discarded, and they are replaced with rigid connections.

The final step, as alluded to above, is to determine the weights of both neuron-neuron and sensor-neuron connections within the CTRNN. One additional CPPN output, w, is used to determine these connection weights. For the sensor-neuron connections it is necessary to distinguish between the different sensors that may exist at the same location. For this purpose additional CPPN inputs are utilized. There are four additional inputs: one for each sensor that are set to 1 when determining connectivity from that sensor, and set to 0 at all other times.

[1] A "leaf" sphere is one which no other sphere has been attached to, and therefore is only connected to its parent. It is a leaf of the morphological tree.

Each pair in {sensors} x {neurons} is queried by providing the specific sensor input as just described and by providing the coordinates of the midpoint between the given sensor and neuron's locations. Similarly each pair in {neurons} x {neurons} is queried by providing the coordinates of the midpoint between the two neurons' locations. In this way a monolothic CTRNN controller is created with links from all sensors to all neurons and links between all neuron pairs (including self feedback loops), though links can be effectively eliminated by having weights near zero.

As opposed to being used as an encoding of robot morphologies, CPPNs have been commonly employed to encode the connection weights of neural networks via the HyperNEAT algorithm (Stanley et al. 2009). Customarily the neurons for which connection weights are to be determined are presented to the CPPN as existing on independent hyperplanes with two sets of Cartesian coordinates given as inputs. Even though the method just described places restrictions on the weight distributions that can be represented by the CPPN (e.g. only allowing for symmetric connections) preliminary experimentation demonstrated that this was not detrimental to performance in this domain. Accordingly the midpoint method, with its need for only one set of Cartesian coordinates, is utilized to avoid adding even more complexity to the already complex CPPNs.

6.2.4 Selecting desirable robots

The first goal of this paper is to demonstrate that the methods presented above are capable of evolving robots with closed-loop CTRNN controllers for a given task. In particular the task chosen for investigation is maximum directed displacement of the robot in a fixed amount of time.

To select for this property, a fitness, f is calculated as the sum of two parts $f = r + d$. This function first rewards robots for possessing the components necessary to displace themselves and sense a target object they are navigating towards (the r term). Then, if the robot possesses these necessary components, the robot is placed in a physical simulator[2] for a set amount of time and the second part of the fitness function (the d term) is calculated.

The r term is used to guide evolution towards potentially successful solutions prior to simulation and its accompanying overhead, and was formulated based on previous experimentation. It begins with a value of 1. If the robot possesses any sensors then r is incremented by 1. If the robot possesses any joints then r is incremented by 1. If the robot possesses any inter-neuron connections then r is incremented by 1. Finally, if

[2]Simulations are conducted in the Open Dynamics Engine (`http://www.ode.org`), a widely used open source, physically realistic, simulation environment

the robot possesses any sensor-neuron connections then r is incremented by 1. If the robot has earned all of these reward points, and one of the robot's sensors is a distance sensor (which allows it to remotely sense the target object) then the robot will be sent to the simulator and r is incremented by an additional 10 to further reward potentially valid solutions.

If the robot is sent to the simulator it is allowed to act for a set amount of time or until an early-termination condition is met. At the conclusion of the simulation, the d term is calculated as $d = d_{\text{init}} - d_{\text{final}}$ where d_{init} is the robot's initial distance to the target object and d_{final} is the robot's distance to the target object at the end of the simulation.

The first of the early termination conditions is simply to save computational resources. If all of a robots parts have completely stopped moving then the simulation is stopped and d_{final} is considered to be the robot's distance to the target object at this time. Another condition is used to prevent robots from exploiting simulation faults. There are a number of ways these faults could be avoided such as reducing the step size used for the underlying differential equations within the simulation, but this would lead to increased simulation run times. The technique used here is to throw out any solution where the robot's linear or angular accelerations exceed predefined thresholds. In this case as soon as one of the thresholds is exceeded the simulation is terminated and d_{final} is set to be d_{init} so that $d = 0$.

Finally, there are two additional criteria that need to be met for the d fitness component to be calculated as described. These conditions are used to prevent solutions where the robot moves by rolling on a subset of its components. These solutions tend to be common (since the robots are composed of spheres) but are less interesting than other solutions that may be found and so are considered invalid. At the conclusion of a simulation run any robot that is found to have a subset of its spheres remain in contact with the ground for over 95% of the simulation time is considered to be invalid and d_{final} is set to be d_{init} once again. Also at the conclusion of the simulation the angular velocities of each rigid body component are averaged over time. If, for any of these body components, the magnitude of this average exceeds a pre-defined threshold than the robot is also considered to be invalid and d_{final} is set to be d_{init} once again. This ensures that no component is constantly rotating in the same direction which would be indicative of a robot that is rolling on a subset of its spheres.

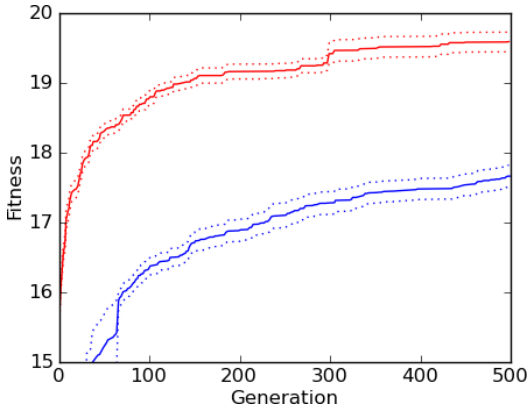


Figure 6.1: Fitness plots for the control and experimental regimes. The experimental regime significantly outperforms the control for the entire 500 generations. Control is shown blue and experimental in red. Solid lines denote mean fitness at each generation, while dotted lines depict +/- one unit of standard error.

6.3 Results

In the previously published works using CPPNs to produce three-dimensional structures and actuated robot morphologies (Auerbach and Bongard 2010b, Auerbach and Bongard 2010a) the CPPNs were all restricted to only using feed-forward connections. That is, recurrent connections within the CPPNs were not allowed. This policy of not allowing recurrent connections is common when using CPPNs and was initially copied here, but the question arises: are recurrent CPPN connections useful in this domain?

In order to answer this question two experimental regimes are formulated. In the first, the **control regime**, recurrent connections are not allowed. In the second, the **experimental regime**, recurrent connections are permitted. In both regimes the CPPNs have the values of their nodes reset prior to every query. Additionally, the CPPNs are updated for a fixed number of iterations (in this case 10) before the output values are retrieved. This update procedure is common when using feed-forward CPPNs and is used with the recurrent CPPNs here in order to avoid the complexities of networks that do not settle down to a steady state (i.e. those that exhibit cyclic or chaotic dynamics).

Each regime consists of 30 independent trials using CPPN-NEAT to evolve robot morphologies and embedded CTRNN controllers for directed displacement as described above. Moreover, all trials are configured to use a population size of 150, and run for 500 generations with each fitness evaluation in the simulator given 2500 time steps. Additionally in all experiments the values T_{matter} and T_{joint} are both fixed at 0.7, r_{init} is set to

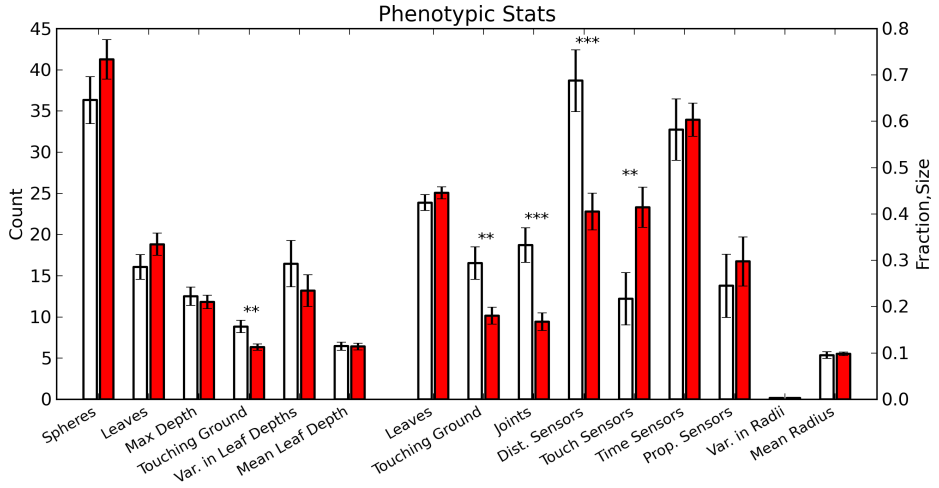


Figure 6.2: Comparison of morphological statistics. This plot compares several statistics between the best of run robots evolved in the control regime (white) and experimental regime (red). The left hand axis is used for the leftmost six pairs while the right hand axis is used for the other pairs. Asterisks denote statistics that are significantly different between the two regimes: * denotes p-values < 0.05, ** denotes p-values < 0.01, and *** denotes p-values < 0.001

0.1, r_{min} is set to 0.05, r_{max} is set to 0.5, M is set to 50, and each sphere of the morphology is restricted to having its center initially located in interval $([-2, 2], [-2, 2], 0)$[3]. These values were chosen based on experimentation to allow for a diverse range of morphologies that could be stably simulated in a reasonable amount of time. Before being placed in the simulator the morphologies are translated vertically such that the largest component is resting on the ground. The CPPN internal nodes are allowed to use the signed cosine, Gaussian, and sigmoid activation functions to allow for repetition, symmetry and variation. All other parameters of the evolutionary algorithm are kept at the default values provided with the C++ implementation of HyperNEAT[4].

Pictures of the top best of run individuals are shown in Fig. 6.4 and videos of their behaviors are available online at `http://www.cs.uvm.edu/~jauerbac/`. Both regimes produce robots that can displace themselves in the desired direction on the order of several body lengths in the allotted evaluation time, however the experimental regime is able to produce robots capable of displacing significantly farther on average then those from the control regime (p-value < 0.001 at the end of 500 generations[5]).

Fitness plots comparing the two regimes are plotted in Fig. 6.1. It can be seen here that the performance difference is apparent at the beginning of the experiments and exists for the entire 500 generations of evolu-

[3] sizes and coordinates are all in arbitrary ODE units

[4] Available at `http://eplex.cs.ucf.edu/hyperNEATpage/HyperNEAT.html`

[5] This and all other p-values reported in this paper are calculated using ttest_ind from the SciPy stats package.

tionary time. Moreover, even when the control regime is allowed to run for twice as many generations (not depicted here) it still does not achieve the performance of the experimental regime.

6.4 Discussion

Since these results demonstrate that including recurrent connections in the evolving CPPNs significantly improves the fitnesses achieved, the question arises as to why this is so. Specifically how are the robots produced from CPPNs with recurrent connections different from those produced from CPPNs without these connections? Are they simply larger or more complex in some way due to the recurrent feedback loops increasing the saturation of the CPPN's outputs? Do they tend to branch out further or use differently sized spheres than the morphologies produced in the control regime? Or, is some other factor critical to their success?

In order to answer these questions a variety of statistics relating to the body plans of evolved morphologies are computed, and plotted in Fig. 6.2. While the morphologies from the experimental regime do tend to use slightly more spheres and have slightly more leaves, which would be indicative of the feedback loops increasing output nodes' saturation, neither of these differences is statistically significant. Moreover it cannot be the case that all outputs have this increased saturation because the experimental regime evolves robots with significantly fewer joints and significantly fewer distance sensors. Therefore some other explanation is warranted.

What if one considers the size of the spheres in the evolved morphologies? The mean radii across all best of run individuals from both regimes is nearly identical and the mean variance of sphere sizes within the individual morphologies is also similar between the two regimes. Additionally if the maximum number of spheres separating any sphere from the root sphere (max depth) is considered, it is also shown not to differ significantly from one regime to another. So, the increased performance is not caused by having morphologies that branch out further, have larger spheres or a greater spread of sphere sizes.

What else may be causing the robots in the experimental regime to outperform the control runs by such a wide margin? The robots from the experimental regime do tend to have significantly fewer components that come into contact with the ground (both in total number and in fraction of all body components), and as mentioned above the experimental regime does produce robots with significantly fewer actuated joints. It

is possible that this is indicative of the experimental regime producing more efficient control strategies, but what in the encoding enables it to do so is still unclear.

Additionally, when compared to spheres belonging to robots produced by the control regime, the spheres belonging to robots produced by the experimental regime are significantly less likely to contain distance sensors while at the same time being significantly more likely to contain touch sensors. This is interesting, because in this directed displacement task, where the target object is always in the same location, distance sensors are not necessary (though having at least one is required by the fitness formulation). So having fewer distance sensors is a clue that the experimental regime can better restrict its complexity in useful ways. Similarly touch sensors can be useful for producing dynamic behavior so the fact that the experimental regime tends to use more of these is a clue that it can complexify the morphologies as needed.

Another possibility is that the experimental regime has found ways to produce morphologies that, while they do not significantly differ in many of the structural statistics just presented, do have structural differences not captured by these statistics. Perhaps answers can be gleaned by visually inspecting the evolved morphologies. Fig. 6.4 shows images of the most fit, best-of-run individuals from each regime. It appears (although is not yet confirmed) that the robots produced by the experimental regime create more fractal like structures which is made possible by the inclusion of the recurrent CPPN connections. Perhaps this a key to their success. Fractal patterns are common in biological organisms (Ball 2009) and it has been proposed that they would be useful in robotics (Moravec et al. 1996). Further investigation is needed to determine if this is indeed the case.

What about the evolved CPPNs themselves? Is there some structural property of the CPPNs that may provide an explanation of the experimental regime's success? Fig. 6.3 compares a few relevant genotypic statistics. CPPNs evolved in the experimental regime tend to have fewer nodes and correspondingly fewer hidden nodes with each of the possible activation functions. This possibly signifies that these CPPNs can more easily succeed without the added complexity of additional hidden nodes. A much greater disparity exists in the number of links (with CPPNs from the experimental regime having more) but this is not surprising considering the much greater number of links that are possible when recurrence is allowed. Perhaps a more meaningful statistic is the number of forward links, or those that are possible in both regimes. Here, once again, CPPNs from the experimental regime have significantly fewer forward links. This again suggests that these CPPNs can succeed without adding as much unneeded complexity.

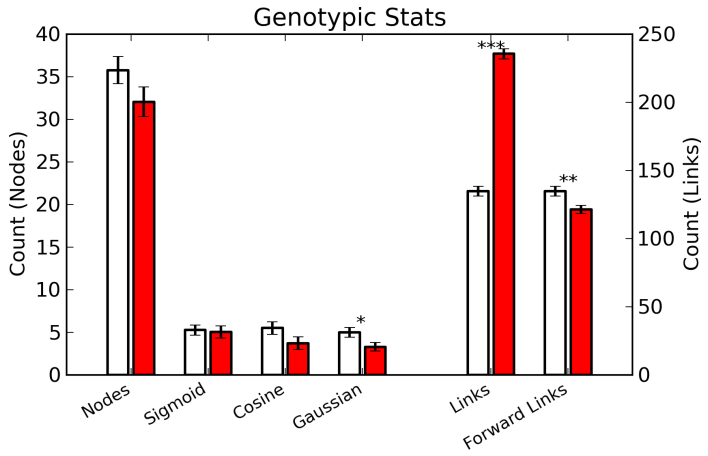


Figure 6.3: Comparison of genotypics statistics. This plot compares several genotypic statistics between the best of run CPPNs from the control regime (white) and experimental regime (red). The left hand axis is used for the number of nodes and number of hidden nodes with a given activation function. The right hand axis is used for the number of links.

One last hypothesis is that recurrent CPPNs exist in a search space that is more conducive to optimizing robot morphologies compared to their feed-forward counterparts. If this is the case then through the course of an evolutionary trial it would be more likely that a best of generation champion would be supplanted by an individual that produces a topologically different morphology. This is indeed the case: when a new champion arises in experimental regime trials 96.99% of the time its morphology is topologically different from the previous champ compared to this happening only 59.92% of the time in the control regime, and this difference is significant ($p < 0.001$). Moreover when this does happen the magnitude of fitness improvements is on average significantly greater in the experimental regime (0.559 vs. 0.188, $p < 0.001$), and even when the new champion is a robot with the same morphological topology as the old champ the magnitude of fitness improvements is still greater in the experimental regime (0.255 vs. 0.100 $p < 0.05$) indicating that the recurrent CPPNs are not only more suited to optimizing morphology, but are more suited to optimizing controllers as well. Therefore, the space of robots encoded by recurrent CPPNs is more evolvable, however determining exactly why this is the case will require additional examination.

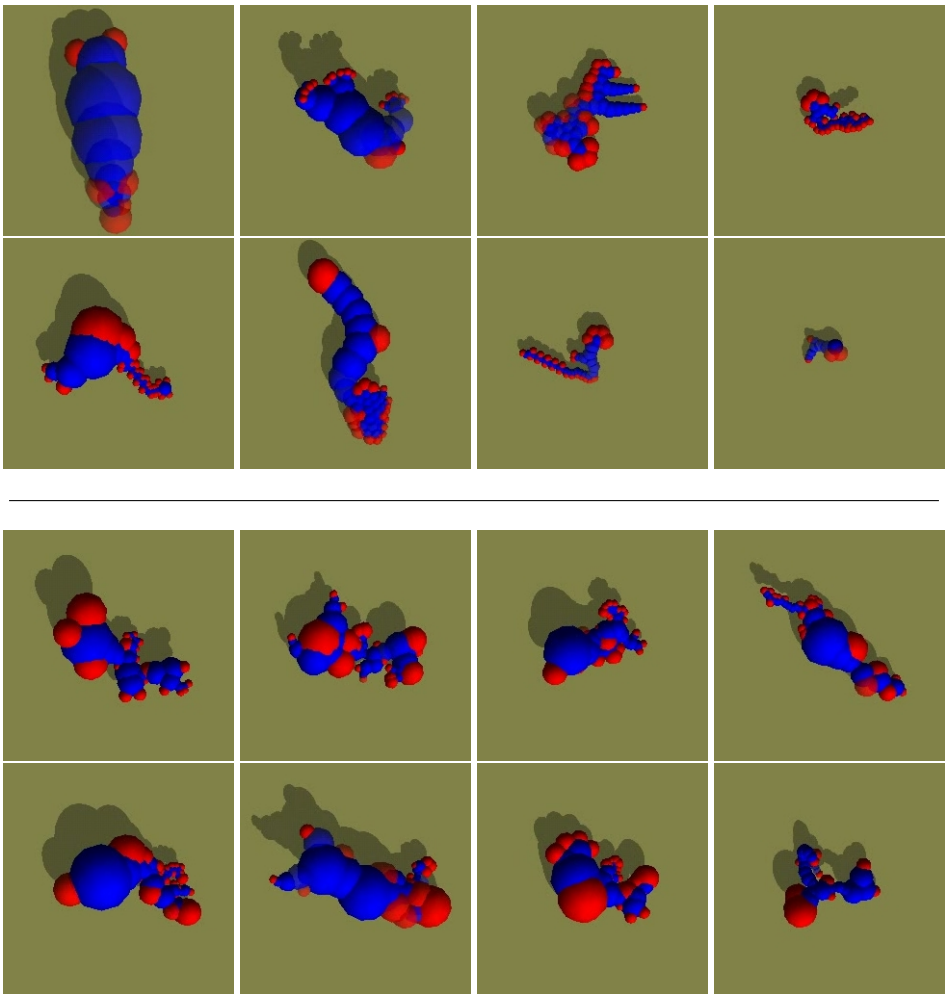


Figure 6.4: The Zoo: pictures of evolved robots. The top eight best of run individuals from the control regime (top) and experimental regime (bottom) are shown. Leaf spheres are colored red while all other spheres are colored blue. Videos of these robots in action are available at `http://www.cs.uvm.edu/~jauerbac`

6.5 Conclusion

This paper has presented a method for evolving complete robots including their physical topologies, sensor placements, and embedded, closed-loop, neural network control policies using Compositional Pattern Producing Networks as the underlying generative encoding. It demonstrated how this method works on a sample task and showed how including the possibility of recurrent connections within the underlying CPPNs significantly improves the fitnesses achieved on that task.

This result poses the question of why including recurrent connections allows for the creation of more successful robots. Several hypotheses were presented attempting to elucidate the matter. Some of these hypotheses were able to be discarded by analyzing statistics of the evolved morphologies and their underlying CPPNs, while others could not be rejected without additional information. Future work will aim to seek out additional answers to this question in the hopes of using the knowledge gained to further improve the presented algorithm.

Additionally, going forward, the authors plan to tackle more complex tasks such as photo-taxis and object manipulation to test whether the methods used in this work will continue to be successful and if the utility of including recurrent outputs extends to these other domains.

6.6 References

Adamatzky, A., M. Komosinski, and S. Ulatowski (2000). Software review: Framsticks. *Kybernetes: The International Journal of Systems & Cybernetics 29*(9/10), 1344–1351.

Anderson, M. (2003). Embodied Cognition: A field guide. *Artificial Intelligence 149*(1), 91–130.

Auerbach, J. E. and J. C. Bongard (2010a). Dynamic Resolution in the Co-Evolution of Morphology and Control. In *Artificial Life XII: Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems*.

Auerbach, J. E. and J. C. Bongard (2010b). Evolving CPPNs to grow three-dimensional physical structures. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Ball, P. (2009). *Branches: Nature's Patterns: A Tapestry in Three Parts*. Oxford University Press.

Beer, R. D. (2006). Parameter space structure of continuous-time recurrent neural networks. *Neural Comp. 18*(12), 3009–3051.

Beer, R. D. (2008). The dynamics of brain-body-environment systems: A status report. In P. Calvo and A. Gomila (Eds.), *Handbook of Cognitive Science: An Embodied Approach*, pp. 99–120. Elsevier.

Bongard, J. and R. Pfeifer (2001). Repeated structure and dissociation of genotypic and phenotypic complexity in Artificial Ontogeny. *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO)*, 829–836.

Bongard, J. C. (2002). Evolving modular genetic regulatory networks. In *Proceedings of The IEEE 2002 Congress on Evolutionary Computation (CEC2002)*, pp. 1872–1877.

Bongard, J. C. and R. Pfeifer (2003). Evolving complete agents using artificial ontogeny. In *Morpho-functional Machines: The New Species (Designing Embodied Intelligence*, pp. 237–258. Springer-Verlag.

Brooks, R. (1999). *Cambrian intelligence*. MIT Press Cambridge, Mass.

Clune, J., B. Beckmann, C. Ofria, and R. Pennock (2009). Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computing*, pp. 2764–2771.

Clune, J., R. T. Pennock, and C. Ofria (2009). The sensitivity of HyperNEAT to different geometric representations of a problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

Dellaert, F. and R. Beer (1994). Toward an evolvable model of development for autonomous agent synthesis. *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*.

Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. *Procs. of the Fourth European Conf. on Artificial Life*, 205–213.

Harvey, I., P. Husbands, D. Cliff, A. Thompson, and N. Jakobi (1997). Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems 20*, 205–224.

Hornby, G. S. and J. B. Pollack (2001a). Body-brain co-evolution using L-systems as a generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, pp. 868–875. Morgan Kaufmann.

Hornby, G. S. and J. B. Pollack (2001b). Evolving L-systems to generate virtual creatures. *Computers and Graphics 25*(6), 1041–1048.

Komosinski, M. and A. Rotaru-Varga (2002). Comparison of different genotype encodings for simulated three-dimensional agents. *Artif. Life 7*(4), 395–418.

Lipson, H. and J. Pollack (2000). Automatic design and manufacture of robotic lifeforms. *Nature 406*, 974–978.

Lund, H. H., J. Hallam, and W. Lee (1997). Evolving robot morphology. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*, Piscataway, NJ, USA. IEEE Press.

Mautner, C. and R. Belew (2000). Evolving robot morphology and control. *Artificial Life and Robotics 4*(3), 130–136.

Moravec, H., J. Easudes, and F. Dellaert (1996). Fractal branching ultra-dexterous robots (bush robots). Technical report, NASA Advanced Concepts Research Project. PR-Number 10-86888.

Nolfi, S. and D. Floreano (2000). *Evolutionary Robotics: The Biology,Intelligence,and Technology*. Cambridge, MA, USA: MIT Press.

Pfeifer, R. and J. Bongard (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press.

Sims, K. (1994). Evolving 3D morphology and behavior by competition. *Artif. Life 1*(4), 353–372.

Stanley, K., D. D'Ambrosio, and J. Gauci (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life 15*(2), 185–212.

Stanley, K. and R. Miikkulainen (2003). A taxonomy for artificial embryogeny. *Artificial Life 9*(2), 93–130.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines 8*(2), 131–162.

Stanley, K. O. and R. Miikkulainen (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation 10*(2), 99–127.

Chapter 7

On the Relationship Between Environmental and Morphological Complexity in Evolved Robots

The principles of embodied cognition dictate that intelligent behavior must arise out of the coupled dynamics of an agent's brain, body, and environment. While the relationship between controllers and morphologies (brains and bodies) has been investigated, little is known about the interplay between morphological complexity and the complexity of a given task environment. It is hypothesized that the morphological complexity of a robot should increase commensurately with the complexity of its task environment. Here this hypothesis is tested by evolving robot morphologies in a simple environment and in more complex environments. More complex robots tend to evolve in the more complex environments lending support to this hypothesis. This suggests that gradually increasing the complexity of task environments may provide a principled approach to evolving more complex robots.

7.1 Introduction

According to the principles of embodied cognition intelligent behavior arises out of the coupled dynamics between an agent's body, brain and environment (Brooks 1999, Anderson 2003, Pfeifer and Bongard 2006, Beer 2008). This suggests that the complexity of an agent's control policy (brain) and physical body (morphology) should scale proportionally to the complexity of its task environment. This link between control and morphology has been studied (Paul 2006), however the relationship between environmental complexity and morphological complexity is not well understood.

Evolutionary robotics (ER) (Harvey et al. 1997, Nolfi and Floreano 2000), the application of evolutionary algorithms to the design and optimization of robot control policies and/or morphologies, provides a framework for investigating this relationship. While most evolutionary robotics projects have restricted themselves to optimizing control strategies for human designed or bio-mimicked robot body plans, evolutionary algorithms may also be used to design complete robots: physical morphologies in addition to control policies. Evolving morphology in addition to control has the advantage of being able to discover body plans uniquely suited to a machine's given task environment rather than suffering from the design biases of human engineers.

This idea of allowing an evolutionary algorithm to control both the morphologies and controllers of simulated machines in virtual environments to produce adaptive behavior was first introduced by Sims (Sims 1994). Sims' was followed by other studies (e.g. (Lund et al. 1997, Adamatzky et al. 2000, Mautner and Belew 2000, Lipson and Pollack 2000, Hornby and Pollack 2001, Komosinski and Rotaru-Varga 2002, Stanley and Miikkulainen 2003, Eggenberger 1997, Bongard and Pfeifer 2001, Bongard 2002, Auerbach and Bongard 2010a, Auerbach and Bongard 2011)) which also explored evolving both the morphologies and control policies of robots in virtual environments. These studies varied in a number of meaningful ways including their underlying genetic encodings, the parts with which the robots were constructed, the evolutionary algorithms employed, and the tasks investigated. However, by far the most commonly investigated task in this line of research has been locomotion over flat terrain: how far a robot is able to displace itself over flat ground in an allotted amount of time.

While interesting results have come from investigating this task it suffers from its simplicity. Relatively simple morphologies of just a few cuboids or spheres are all that is needed to be successful. However, it is of great interest how morphological complexity scales in more complex task environments, therefore ad-

ditional task environments must be investigated. Previous studies have looked at evolving robots in more challenging task environments (e.g. (Lassabe et al. 2007)), but because these studies used body plans composed of cuboids, like Sims' system, there was a low ceiling on the maximum complexity of their evolved morphologies.

The current study aims to investigate the relationship between environmental and morphological complexity in a more principled way in order to test the hypothesis that the morphological complexity of a robot increases commensurately with the complexity of its task environment. While it draws inspiration from the previous studies mentioned above, the evolutionary system presented here has several advantages which make it better suited to studying this issue.

One advantage concerns the genetic encoding employed and the manner in which robot morphologies are modeled. As has been demonstrated in the past (Hornby and Pollack 2001, Komosinski and Rotaru-Varga 2002) generative and developmental encodings offer demonstrable benefits over direct encodings for evolving robot morphologies. Accordingly, the morphologies in this study are created from a specific generative encoding that has been shown to possess a host of advantages over other encodings: Compositional Pattern Producing Networks (CPPNs) (Stanley 2007)[1]. This is similar to what was done in (Auerbach and Bongard 2010a, Auerbach and Bongard 2011), however in lieu of building robots out of spherical components via a growth procedure as is done in (Auerbach and Bongard 2010a, Auerbach and Bongard 2011) morphologies are instead created out of triangular meshes (trimeshes) based on sampling a CPPN output at regular intervals over a region of space. The flexibility of trimeshes allows for the creation of a greater diversity of morphologies than is possible with cuboids or spheres (see Figures 7.1 and 7.4 for examples of morphologies evolved with the current system).

Another advantage of the current system is the genetic algorithm employed. Many advances have been made in developing more successful evolutionary algorithms since Sims' work, which should allow for searching the space of robot morphologies more effectively. Specifically, in this research, populations of CPPN genomes are evolved using CPPN-NEAT: an extension of the widely used NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen 2002) algorithm. Some of the advantages of CPPN-NEAT are presented in the next section.

[1]For more on the specific advantages of CPPNs as a genetic encoding the reader is directed to (Stanley 2007, Stanley et al. 2009, Clune et al. 2009, Clune et al. 2009, Auerbach and Bongard 2010b, Auerbach and Bongard 2010a, Auerbach and Bongard 2011).

A final advantage worth mentioning is the vast amount of computational resources that many modern researchers have access to. These resources are necessary to run large numbers of physical robotics simulations at small enough step sizes to produce physically plausible results. All the experiments presented in this paper are carried out on a 7.1 teraflop supercomputing cluster. Without access to such a distributed computing system one single evolutionary run from one single experiment would take multiple days to complete on a standard personal computer. But, when using the cluster, an entire experiment (of 100 runs) can be run in less than one day thus allowing for experimentation with a large number of environments within which enough runs may be conducted to produce statistically significant results.

The remainder of this paper is organized as follows: the next section further describes the CPPN encodings used, describes how they evolve and how they produce actuated robots. A description of the different simulated environments in which robots are evolved then follows. Next, results are presented which capture how different environments affect the complexity of the robot morphologies that evolve inside them. This is followed by a discussion of how the complexity of a robot body plan may be calculated using geometric properties and information theoretic measures. These techniques are then applied to the evolved robot body plans and relationships between environmental and morphological complexity are examined. The paper finishes with concluding remarks and a discussion of how the ideas presented in this paper may be extended in future work.

7.2 Methods

7.2.1 CPPNs

Compositional Pattern Producing Networks (CPPNs) (Stanley 2007) are a form of artificial neural network (ANN). However, CPPNs differ from traditional ANNs in several important ways. Unlike traditional ANNs where every internal node has the same activation function (such as a sigmoid or a step function) CPPN nodes can take on one of several activation functions from a predefined set. This function set often includes functions that are repetitive such as sine or cosine as well as symmetric functions such as Gaussian, thus allowing for motifs seen in natural systems: symmetry, repetition, and repetition with variation. Additionally CPPNs are often used as a generative system to encode some other object of interest e.g. pictures (Secretan et al. 2011), 3D structures (Auerbach and Bongard 2010b, Clune and Lipson 2011), robot morphologies

(Auerbach and Bongard 2010a, Auerbach and Bongard 2011) or traditional ANNs (Stanley et al. 2009), as opposed to being employed directly as a control architecture as ANNs typically are. Here CPPNs are used as such a generative encoding to produce actuated robot body plans. A more in depth description of CPPNs is beyond the scope of this paper; the reader is referred to (Stanley 2007) for further details.

7.2.2 Evolutionary Algorithm

In this study CPPN-NEAT (Stanley 2007) is the algorithm used to evolve CPPNs. CPPN-NEAT is an extension of the state of the art NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen 2002) method of neuro-evolution. NEAT and by extension CPPN-NEAT begins with small networks (those without any internal or hidden nodes) and gradually increases the complexity of the networks over time through the addition of new nodes and links. Additionally the population is divided into "species" for the purpose of promoting genotypic diversity and allows novel structural innovations time to mature. This systematic way of increasing network complexity as needed should lend itself well to studying how morphologies increase in complexity when evolving inside different environments. A more thorough description of the NEAT and CPPN-NEAT algorithms also falls outside the bounds of this paper, so the reader is directed to (Stanley and Miikkulainen 2002, Stanley 2007) for additional details.

7.2.3 Building Robots from CPPNs

In previous studies (Auerbach and Bongard 2010a, Auerbach and Bongard 2011), robots were constructed out of spherical components from evolving CPPNs by means of an iterated growth procedure. This procedure involved starting at a specific initial point and attaching spheres to grow outwards by means of querying the CPPN genome locally and placing newly created spheres in a priority queue whereby they could be selected as attachment points for additional spheres. This process would repeat until a complete robot was grown.

While promising results were produced by the system presented in those papers it has several drawbacks. In many cases the additional indirection added by the growth procedure prevents desirable features of the CPPNs' outputs – such as symmetry and repetition – from being realized in the resulting morphologies. Moreover, while spheres are easy to physically simulate due to their single points of contact such that morphologies with a small number of spheres can be cheaply simulated, the computational costs become too

large (even on a cluster) when trying to model sufficiently complex physical shapes with spheres. Because of these considerations an alternative method is employed in this work.

In lieu of the growth procedure just described the current study employs a voxel based method to create morphological components out of triangular meshes (trimeshes) similar to what is done for the creation of 3D shapes in (Clune and Lipson 2011). A regular grid is placed over a region of 3D-space which defines the presence of voxel locations. In the current work this region extends from -1 to 1 (inclusive) in each dimension and grid lines are placed at intervals of 0.2. This yields a total of 11 grid lines in each dimension for a total of 1331 voxels.

A candidate CPPN is iteratively queried with the (x, y, z) Cartesian coordinates at every voxel location except for the extrema in each direction. Voxel locations that exceed a predefined output threshold (0.5 in this case) are considered to contain matter, while those that do not exceed this threshold are considered to be devoid of matter. All voxels lying on one of the extrema ($|x| = 1$ or $|y| = 1$ or $|z| = 1$) are given output value 0 to ensure that the final triangular meshes have completely enclosed surfaces. Once the CPPN has been queried for every voxel location the Marching Cubes algorithm (Lorensen and Cline 1987) is employed to create triangular meshes from the underlying voxel data. Specifically an enclosed triangular mesh is created for each connected voxel component which defines the exterior surface of a single physical shape. It is these triangular meshes which are sent to the physics simulator where they define the exterior surface of a solid object and are imbued with mass. As far as the authors are aware this is the first instance of physically simulating evolved, rigid body robots composed of triangular meshes.

Since the purpose of this study is to investigate how different task environments affect the shapes of evolved morphologies, a number of simplifications are used in order to concentrate on the physical shapes of the evolved robots and control for other factors that may influence their performance. From the multiple enclosed trimesh components that could be produced when querying a single CPPN only one of these (the largest in terms of number of triangles) is used in the resulting robot. This single component is copied and reflected across the x-axis. The resulting components (the original and its mirror image) are then spread apart by 0.2 units and a capsule of this length is placed between them such that it connects their two closest points. The two trimesh components each connect to this capsule by means of a hinge joint. These joints have rotation normals of $(1, 0, 0)$ and $(0, 0, -1)$ such that the joints rotate through the robot's coronal and sagitall planes respectively. Reflecting and copying a single component like this ensures that all robots have the same

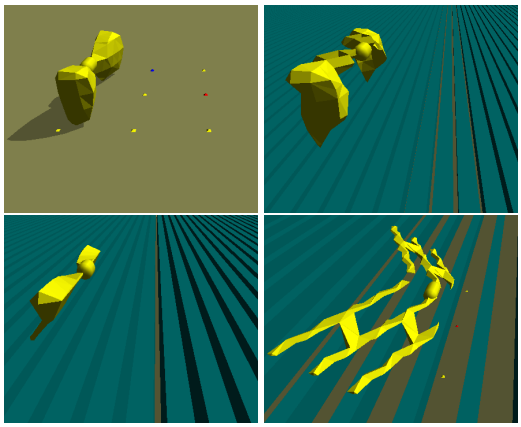


Figure 7.1: Evolved robots and their environments. The control environment and a sampling of the experimental environments are shown with robots that evolved to locomote successfully in each. The ground is a high friction surface, while the blue "blocks of ice" have very low friction. To view videos of these robots in action visit `http://tinyurl.com/GECCO2012-Videos`

degrees of freedom and ensures that the robots are all bilaterally symmetric (which should make locomotion easier) while at the same time it allows for a vast number of different morphologies due to the flexibility of the trimesh model.

The two degrees of freedom of each robot are actuated by means of coupled oscillators. Each of the two oscillators is parameterized by several parameters: amplitude, period, and phase shift. These six parameters (three parameters apiece for each of the two joints) are directly encoded in the genome of the evolving robots as floating point numbers so that the genome is in actuality a CPPN plus a six dimensional floating point array. These floating point numbers are recombined and mutated in exactly the same manner as CPPN link weights except that since every individual possesses these parameters crossover is possible in all instances of sexual reproduction. Values for these parameters are constrained to predefined ranges: amplitude, $a \in \left[\frac{\pi}{4}, \frac{3\pi}{4}\right]$ (so that the hinge rotates between $-a$ and a radians), period $\in [250, 1500]$ simulation time steps (or equivalently $[2, 12]\%$ of the total evaluation time) and phase shift $\in [-1, 1]$ periods. Each parameter has a mutation probability of 0.1, which was chosen experimentally.

Encoding the control parameters in this fashion is done to keep the controllers as simple as possible so that fitness is primarily dictated by the physical form of the robots while at the same time allowing for diverse enough behavior so that the robots can succeed in the different task environments.

7.2.4 Selecting desirable robots

The focus of this study is on how varying the complexity of task environments affects the complexity of evolved robot morphologies. Towards this aim a simple task is chosen which can be accomplished with more or less difficulty in a variety of environments. Specifically, like in previous work, the task investigated here is maximizing directed displacement in a fixed amount of time, though this is done across a range of environments and not just on flat ground.

A candidate robot morphology (triangular mesh) and accompanying control parameters are sent to a physics simulator[2] and allowed to act for a fixed number of simulation time steps. Since trimeshes can be arbitrarily shaped and, unlike spheres, may simultaneously contact the environment at several points it is necessary to use a much smaller step size than has been used in previous work in order to get physically realistic behavior. Specifically, a step size of 0.001s is used in this work. Because of this smaller step size a proportionally larger number of time steps are needed to achieve the same effective simulation length. Here robots are evaluated for $T = 12500$ time steps.

After the robot has completed its time in the simulator its fitness is calculated. How exactly this fitness is calculated takes some care, because evolution often finds ways to "cheat" naïve fitness functions especially when the task environment is difficult. For example, if fitness only considers the positions of the robot's center of mass, C, and takes fitness as $C(T)_x - C(0)_x$ where $C(t)_x$ is the x-coordinate of the robot's center of mass at time t and T is the simulation length then in environments where locomotion is difficult evolution will tend to find solutions where C is initially raised far off the ground so that its displacement can be maximized by falling forward. This is a local optimum in this fitness landscape. Similarly, if one tries to eliminate this cheating by only considering the trailing point of the robot so that fitness is $\min p(T)_x - \min p(0)_x$ where $\min p(t)_x$ is the smallest x-coordinate across all points on the robot at time t falling forward can still be an effective solution (and is still a local optimum) in difficult environments if morphologies are created which have backwards protrusions and thus make $\min p(0)_x$ as small as possible.

In light of these considerations the fitness employed in all environments in this research is $\min p(T)_x - \max p(0)_x$. With this fitness function falling forward will not be rewarded because the maximum fitness that

[2]Simulations are conducted in the Open Dynamics Engine (`http://www.ode.org`), a widely used open source, physically realistic, simulation environment.

can be achieved by pivoting about a single point will be 0 and so a robot must actually displace its whole body forward to be rewarded.

7.2.5 Exploring environments

As mentioned previously the goal of this study is to investigate how varying the complexity of task environments affects the complexity of evolved robots. To accomplish this goal, robots are evolved in a range of environments with tunable parameters that can effectively increase or decrease the difficulty of the task. For each environment investigated 100 independent evolutionary runs of CPPN-NEAT are run for 500 generations with a population size of 150. The implementation of CPPN-NEAT, the parameter settings, and the CPPN activation functions are the same as those used in (Auerbach and Bongard 2011) except for the addition of the floating point array encoding the control parameters, and an improved selection mechanism[3].

The first environment in which robots are evolved in is flat, high friction ground similar to previous work. The robots evolved in this simple environments are considered control cases to compare with robots evolved in other environments. Subsequent environments are more complex: they all consist of an infinite series of low friction rectangular solids ("blocks of ice") over which a robot must locomote. These "ice blocks" are constructed such that it is impossible for a robot to gain purchase by moving over their upper surfaces but must instead reach into the gaps between the blocks to propel themselves forward. This requires the evolution of morphologies with appropriate physical forms. These "icy" environments vary according to two parameters: the height of the blocks and the spacing between the blocks. Each of these parameters varies from 0.025 units to 1.6 units exponentially for a total of $7 * 7 = 49$ different environments. The exponential scaling is used in order to cover a range of parameters which produce qualitatively different environments. Figure 7.1 shows a sampling of these environments and robots that evolve inside them.

7.3 Results

After completing the 100 run in the control environment and another 100 runs for each of the 49 experimental environments (for a total of $50 * 100 = 5000$ evolutionary runs) the most obvious question becomes: how dif-

[3]The authors were made aware through personal correspondence of a bug in the selection mechanism in previous versions of the HyperNEAT C++ distribution. The code was patched to fix this bug (and thus behave as described in the literature) before the current experiments were run.

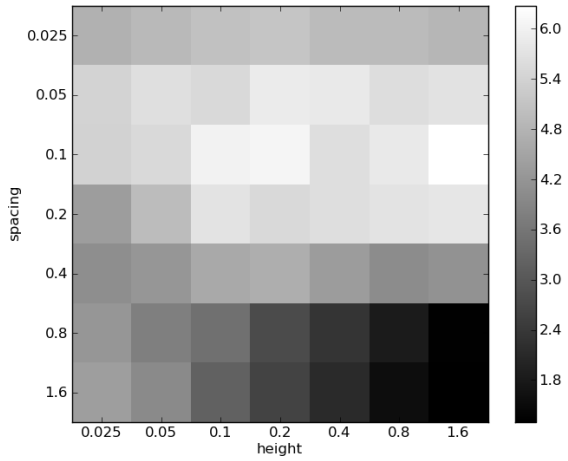


Figure 7.2: Mean distance achieved in each environment. This plot shows the mean distance achieved (in arbitrary ODE units) by the best individual in the final generation taken across the 100 independent runs in each of the 49 experimental environments. For comparison the mean distance achieved from the 100 independent runs in the control environment is 5.09 units.

ficult are these different experimental environments? Or, put another way, how successful is this evolutionary system at producing locomoting robots in each of these environments?

Figure 7.2 shows the mean distance that the best of run individuals are able to locomote (taken across the 100 independent runs) in each experimental environment. This figure demonstrates that there is a clear relationship between these environmental parameters and the difficulty of the task. Specifically, starting in the lower right of this matrix where both the spacing and the height of blocks are large the task becomes very difficult and the robots all become stuck in the gaps unable to successfully locomote. Keeping the spacing constant and decreasing the block height gradually makes the task easier as the robots are able to navigate over these smaller blocks and therefore displace far enough to be considered successfully locomoting. Once the height has been reduced to 0.025 units the blocks are so small that the environment becomes very similar to flat ground and in fact distances achieved by robots in the lower left environments are not significantly different from those of the control environment, nor are the morphologies in this environment significantly different from those of the control environment (see below).

As the spacing between the blocks is reduced the robots are no longer able to behave as they would on flat ground, but instead must find ways to move along the tops of the blocks while finding means of gaining purchase by reaching into the gaps. The height of the blocks loses importance in this part of the parameter space but still has an effect (though opposite to when the spacing is large). Here the general pattern is for taller

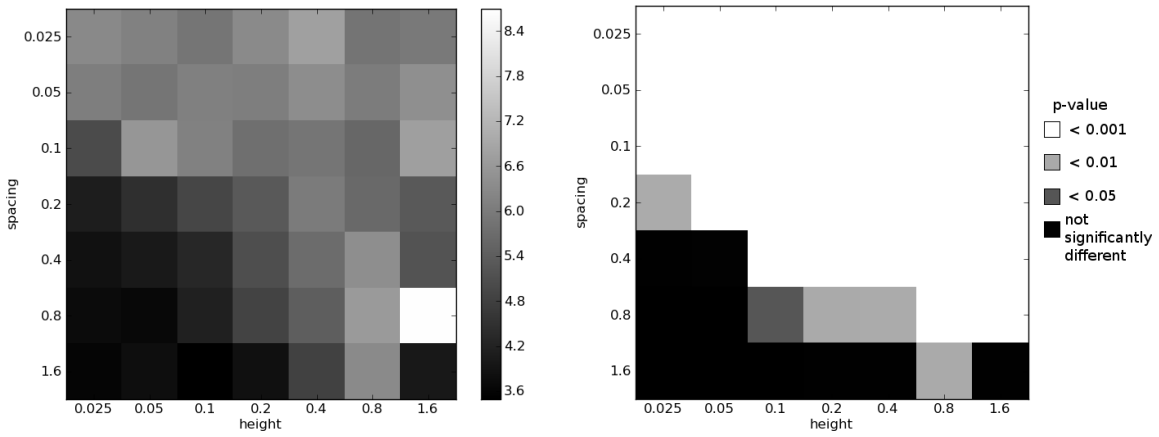


Figure 7.3: How space filling the evolved morphologies are. *Left*: Mean ratio between volume of morphology's AABB and volume of morphology itself for each of the experimental environments. The best of run robots from the control experiment have a mean of 3.58 for this ratio, similar to the black square in this plot. *Right*: Significance of the difference of this ratio in each experimental environment compared to the control environment. The ratio is significantly greater (morphologies are significantly less space filling) on average in the vast majority of experimental environments. There are no experimental environments in which this ratio is significantly smaller than that of the control. All p-values calculated using the Mann-Whitney U test.

blocks to make the task easier, probably because taller blocks result in a greater volume of space whereby the robot can reach into the gaps to gain purchase. Finally at the top of the matrix, when the spacing is smallest block height ceases to have an impact as no matter what forms the robots evolve to they can not reach very far into the gaps.

For a better understanding of how the evolved robots behave in each of these environments it is helpful to watch them in action. For this purpose, videos of robots evolved in each environment are available on the web at `http://tinyurl.com/GECCO2012-Videos`.

7.4 Discussion

It is clear that different environments in this parameterization present the evolutionary system with varying degrees of difficulty, but do they also select for different sorts of morphologies? And if so, can these differences be quantified?

One simple way to study this question is to consider how space filling the evolved morphologies are. This can be done by computing the ratio of the volume of a morphology's Axis Aligned Bounding Box (AABB) to the volume of that morphology itself[4].

Figure 7.3 shows the mean values of this ratio, once again taken across the 100 best of run individuals from each experimental environment. Also plotted is how significantly different this ratio is, on average, in each experimental environment when compared to the best of run individuals from the control environment. In the majority of experimental environments this ratio is significantly greater from that of the robots in the control experiment. This demonstrates that these environments do in fact influence the morphologies of the robots which evolve inside them in quantifiable ways: becoming less space filling than those evolved in the control environment for a large portion of the parameter space. Additionally Fig. 7.3 (left) shows how (at least) one aspect of morphology gradually changes as one moves through this environmental parameter space. This lends support to the chosen parameterization being a good one for the purpose of studying how the morphologies of robots are affected by the environment in which they evolve.

It is clear that the morphologies which evolve in these environments vary in quantifiable (and significant) ways across this parameter space. The question now becomes: do some or all of these environments actually select for more complex morphologies than those that evolve to locomote over flat ground?

There are many ways one might think to quantify the complexity of an evolved morphology. Different measures of how space-filling a morphology is such as the AABB ratio presented above or its surface area to volume ratio or measures of how concave a morphology is (such as the ratio of a morphology's volume to that of the convex hull of its points) may all hint at how complex a morphology is. However, each of these measures may be deceived by relatively simple body shapes.

7.4.1 Entropy of curvature

Instead, it is useful to think about the complexity of a body shape in information theoretic terms. One commonly used measure of complexity is Shannon's Entropy (Shannon 1948), which measures the information content of a random variable. Recent work (Page et al. 2003, Sukumar et al. 2008) has demonstrated how notions of Shannon Entropy can be applied to measuring the complexity of a 3D object by considering the curvature of the object as a random variable. In fact, quantifying the complexity of 3D objects in this way has

[4]For simplicity all morphological measures are computed on the single enclosed trimesh object that is produced by Marching Cubes for a CPPN, i.e. the reflected copy of this trimesh and the connecting capsule are not considered

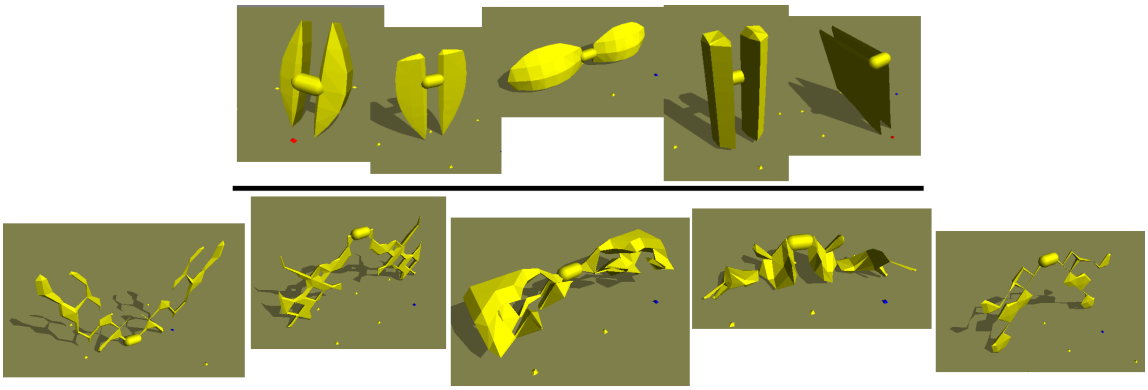


Figure 7.4: Simple and complex morphologies. The five morphologies with smallest (*top*) and largest (*bottom*) values of H_Δ across all best of run individuals from all environments (experimental and control) are shown. The morphologies with high H_Δ values are clearly more complex than those with small H_Δ values.

been shown to strongly correlate with human observers notions of complexity (Sukumar et al. 2008). In the space below the building blocks of computing this measure are presented, and the reader is referred to (Page et al. 2003) and (Sukumar et al. 2008) for more in depth discussions of their theoretical underpinnings.

Given a random variable x with a probability density function (PDF) $p(x)$, entropy H is defined as

$$H = -\sum_i p_i \log p_i \tag{7.1}$$

where $p(x)$ is discretized such that $p_i = \int_{x_{i-1}}^{x_i} p(x)\,\mathrm{d}x$ where the x_is are specific values of x.

But, what is the random variable x on which H will be calculated? Following (Page et al. 2003, Sukumar et al. 2008) x will be a measure of Gaussian curvature of the points on a body shape. Since the body shapes here are built out of triangular meshes the points at which this curvature is non-zero are precisely the vertices of the triangular mesh. Specifically, for each vertex j in a trimesh the angle excess Φ_j is calculated as

$$\Phi_j = 2\pi - \sum_i \phi_i \tag{7.2}$$

where ϕ_i is the internal angle at j of each triangle i of which j is a vertex. This angle excess Φ_j has a direct relationship to the Gaussian curvature at that point (Page et al. 2003). This will be the variable on which entropy is calculated.

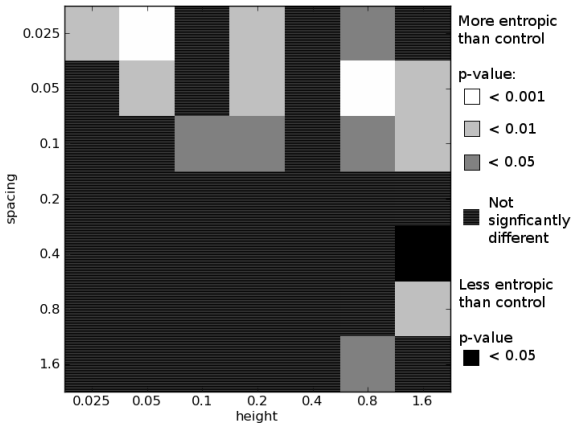


Figure 7.5: Differences in morphological complexity between experimental and control environments. Plot is based on comparing the H_Δ values for best of run individuals in each experimental environment to the H_Δ values for the best of run individuals in the control environment. The more complex experimental environments tend to select for more complex morphologies: there are many experimental environments where significantly more complex morphologies evolve, while only one experimental environment where significantly less complex morphologies evolve. All p-values calculated using the Mann-Whitney U test.

Following the calculation of Φ_j for every vertex a PDF $p(\Phi)$ is estimated by placing the values of Φ_j into discrete bins of uniform width (Δ) and counting the number of Φ_j samples that fall into each bin. This results in a discrete set of probabilities p_i, and Equation 7.1 can be used to arrive at an estimate of entropy that depends on the chosen Δ, denoted here H_Δ[5].

Does H_Δ calculated in this way capture the complexity of evolved morphologies as has been demonstrated in previous work? To answer this question H_Δ is calculated for all 5000 best of run individuals from all environments (experimental and control). Out of those 5000 the five morphologies which have the lowest value for this measure and the five morphologies which have the highest value for this measure are selected. Images of these morphologies are shown in Figure 7.4. Looking at these two sets of morphologies most everyone would agree that those with high H_Δ values appear more complex than those with low H_Δ values. In light of this observation and the previous work in this area it is concluded that H_Δ does a good job of measuring morphological complexity.

[5]The choice of Δ greatly impacts the results of this calculation. If Δ is too large the majority of samples will fall into the same bin and all information is lost. If Δ is too small then the majority of samples will fall into independent bins and H_Δ reduces to a function of the number of vertices n. In general there is no optimum Δ, and since the trimesh morphologies considered here have much fewer vertices than those of (Page et al. 2003) a correspondingly larger bin width must be used. In all calculations presented here a bin width $\Delta = \frac{\pi}{10}$ is used, chosen as a reasonable value by visually inspecting histograms of varying bin widths.

With the knowledge that the complexity of an evolved morphology can be adequately quantified, focus shifts to how the complexity of these morphologies varies from the simple control environment to the more complex parameterized experimental environments. From studying Figure 7.5 one can see that in total the experimental environments tend to select for more complex morphologies than those which evolve in the control environment. Additionally, there is a suggestive pattern across the parameter space where environments in the upper right half of the matrix are much more likely to produce complex morphologies, and this coincides with where the AABB ratios are most significantly different from the control experiment. The upper right of the matrix contains the environments with narrow gaps and higher obstacles, and thus present the most different task from the control environment, so it makes sense that this is where the most different morphologies would evolve. In addition, it makes sense that additional complexity would be needed to succeed in these environments because morphologies with more simple sphere or block like components are unable to reach into the gaps to gain purchase. Therefore evolution must find more complex morphologies to be successful.

7.5 Conclusion

This work has investigated the relationship between environmental and morphological complexity in evolved robots. Using an information theoretic measure of morphological complexity, known to correlate with human perceptions of complexity, it was demonstrated that many complex environments create evolutionary pressures which lead to the evolution of more complex body forms than those of robots evolved in the simple, flat ground environment traditionally investigated. This lends support to the hypothesis that the morphological complexity of a robot should increase commensurately with the complexity of its task environment.

A number of simplifications were made so that the analyses could be focused on the shape of the evolved morphologies. These simplifications included limiting the morphologies to a specific configuration of connectivity with a single connected trimesh reflected and copied and connected via hinge joints to an intermediary capsule, and using an open loop control strategy of coupled oscillators with a small number of evolvable parameters. While the robots evolved in this manner were able to successfully locomote in the majority of environments investigated it would be interesting to investigate how removing these simplifications would affect the presented results. Perhaps with a more sophisticated controller and/or a greater number of degrees of freedom it would be possible to evolve robots which succeed in the most challenging environments or that

are able to succeed in other environments without such an increase in morphological complexity: could increased control complexity supplant the need for increased morphological complexity? However, quantifying the complexity of such robots will require extending the entropy measure presented here to take into account additional factors such as the number and placement of additional degrees of freedom and the complexity of their controller architectures.

The estimation of probability density functions used here in calculating H_Δ could also be improved. As mentioned above, the chosen method depends heavily on selecting a good bin width Δ. Alternative means of estimating a PDF such as kernel density estimation (Silverman 1986) may provide a better means of calculating this measure and will be investigated in future research.

Finally, it will be interesting to see how the morphologies of evolved robots vary in other environments not experimented with in this work. Do certain environments drive an increase in morphological complexity while others drive an increase in complexity of the control strategy or sensory system? Could measuring the complexity of robots while they are evolving be used to inform the evolutionary search process in a meaningful way? Could environments co-evolve along with morphologies much like natural environments change over time, and therefore implicitly drive the evolution of complexity in a more principled way? All of these are fruitful areas for future research.

7.6 References

Adamatzky, A., M. Komosinski, and S. Ulatowski (2000). Software review: Framsticks. *Kybernetes: The International Journal of Systems & Cybernetics 29*(9/10), 1344–1351.

Anderson, M. (2003). Embodied Cognition: A field guide. *Artificial Intelligence 149*(1), 91–130.

Auerbach, J. E. and J. C. Bongard (2010a). Dynamic Resolution in the Co-Evolution of Morphology and Control. In *Artificial Life XII: Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems*.

Auerbach, J. E. and J. C. Bongard (2010b). Evolving CPPNs to grow three-dimensional physical structures. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Auerbach, J. E. and J. C. Bongard (2011). Evolving Complete Robots with CPPN-NEAT: The Utility of Recurrent Connections. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Beer, R. D. (2008). The dynamics of brain-body-environment systems: A status report. In P. Calvo and A. Gomila (Eds.), *Handbook of Cognitive Science: An Embodied Approach*, pp. 99–120. Elsevier.

Bongard, J. and R. Pfeifer (2001). Repeated structure and dissociation of genotypic and phenotypic complexity in Artificial Ontogeny. *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO)*, 829–836.

Bongard, J. C. (2002). Evolving modular genetic regulatory networks. In *Proceedings of The IEEE 2002 Congress on Evolutionary Computation (CEC2002)*, pp. 1872–1877.

Brooks, R. (1999). *Cambrian intelligence*. MIT Press Cambridge, Mass.

Clune, J., B. Beckmann, C. Ofria, and R. Pennock (2009). Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computing*, pp. 2764–2771.

Clune, J. and H. Lipson (2011). Evolving 3D objects with a generative encoding inspired by developmental biology. In *Proceedings of the Eleventh European Conference on Artificial Life (ECAL)*, pp. 144–148.

Clune, J., R. T. Pennock, and C. Ofria (2009). The sensitivity of HyperNEAT to different geometric representations of a problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. *Procs. of the Fourth European Conf. on Artificial Life*, 205–213.

Harvey, I., P. Husbands, D. Cliff, A. Thompson, and N. Jakobi (1997). Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems 20*, 205–224.

Hornby, G. S. and J. B. Pollack (2001). Body-brain co-evolution using L-systems as a generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, pp. 868–875. Morgan Kaufmann.

Komosinski, M. and A. Rotaru-Varga (2002). Comparison of different genotype encodings for simulated three-dimensional agents. *Artif. Life 7*(4), 395–418.

Lassabe, N., H. Luga, and Y. Duthen (2007). A new step for artificial creatures. In *Proceedings of 1st IEEE Conference on Artificial Life (IEEE-ALife 07)*, pp. 243–249. IEEE Press.

Lipson, H. and J. Pollack (2000). Automatic design and manufacture of robotic lifeforms. *Nature 406*, 974–978.

Lorensen, W. E. and H. E. Cline (1987). Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 163–169.

Lund, H. H., J. Hallam, and W. Lee (1997). Evolving robot morphology. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*, Piscataway, NJ, USA. IEEE Press.

Mautner, C. and R. Belew (2000). Evolving robot morphology and control. *Artificial Life and Robotics 4*(3), 130–136.

Nolfi, S. and D. Floreano (2000). *Evolutionary Robotics: The Biology,Intelligence,and Technology*. Cambridge, MA, USA: MIT Press.

Page, D., A. Koschan, S. Sukumar, B. Roui-Abidi, and M. Abidi (2003). Shape analysis algorithm based on information theory. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, Volume 1, pp. I – 229–32 vol.1.

Paul, C. (2006). Morphological computation: A basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems 54*(8), 619–630.

Pfeifer, R. and J. Bongard (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press.

Secretan, J., N. Beato, D. B. D'Ambrosio, A. Rodriguez, A. Campbell, J. T. Folsom-Kovarik, and K. O. Stanley (2011). Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation Journal*, 373–403.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal 27*.

Silverman, B. W. (1986). *Density estimation: for statistics and data analysis*. London.

Sims, K. (1994). Evolving 3D morphology and behavior by competition. *Artif. Life 1*(4), 353–372.

Stanley, K., D. D'Ambrosio, and J. Gauci (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life 15*(2), 185–212.

Stanley, K. and R. Miikkulainen (2003). A taxonomy for artificial embryogeny. *Artificial Life 9*(2), 93–130.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines 8*(2), 131–162.

Stanley, K. O. and R. Miikkulainen (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation 10*(2), 99–127.

Sukumar, S., D. Page, A. Koschan, and M. Abidi (2008). Towards understanding what makes 3d objects appear simple or complex. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2008, Sixth IEEE Workshop on Perceptual Organization in Computer Vision (POCV)*.

Chapter 8

On the Relationship Between Environmental and Mechanical Complexity in Evolved Robots

According to the principles of embodied cognition, intelligent behavior must arise out of the coupled dynamics of an agent's brain, body, and environment. This suggests that the morphological complexity of a robot should scale in relation to the complexity of its task environment. This idea is supported by recent work, which demonstrated that when evolving robot morphologies in simple and complex task environments more complex robot morphologies do tend to evolve in more complex task environments. Here this idea is extended to examining the mechanical complexity of evolved robots. Counter to intuition it is found that the mechanical complexity decreases in more complex task environments.

8.1 Introduction

Proponents of embodied cognition posit that intelligent behavior is a product of the coupled dynamics between an agent's brain, body, and environment (Brooks 1999, Anderson 2003, Pfeifer and Bongard 2006, Beer 2008). Accordingly, the complexity of an agent's brain (control policy) as well as its physical body (morphology) should vary in proportion to the complexity of its task environment. Studying this hypothesis can be approached in several ways. One can investigate the relationship between control and morphology, as was done by Paul (Paul 2006), and one can also study the relationship between task environment and morphology which is less well understood. In recent work (Auerbach and Bongard 2012) we began to investigate this latter relationship by studying how the shape complexity of robot body parts varied when robots were evolved in more or less complex task environments. Here, that work is extended by studying a different aspect of morphological complexity: mechanical complexity, a function of the mechanical degrees of freedom of evolved robots.

The experiments presented in this paper fall within the domain of evolutionary robotics (ER) (Harvey et al. 1997, Nolfi and Floreano 2000). In general ER refers to the practice of employing evolutionary algorithms for the purpose of creating robot control policies and/or morphologies. In the majority of ER studies, control strategies are evolved for human designed or bio-mimicked robot body plans, but it is also possible to use evolutionary algorithms to create complete robots: placing not only robot control strategies under evolutionary control, but the robots' physical morphologies as well. Evolving morphology, in addition to control policy, allows for the discovery of body plans uniquely suited to a machine's given task environment and presents a systematic way to study the relationship between a robot's morphology and the task environment in which it evolved.

The idea of placing both the morphologies and controllers of robots acting in virtual environments under evolutionary control was first introduced by Sims (Sims 1994). Sims' work has been followed by subsequent studies (e.g. (Lund et al. 1997, Adamatzky et al. 2000, Mautner and Belew 2000, Lipson and Pollack 2000, Hornby and Pollack 2001, Komosinski and Rotaru-Varga 2002, Stanley and Miikkulainen 2003, Eggenberger 1997, Bongard and Pfeifer 2001, Bongard 2002, Auerbach and Bongard 2010a, Auerbach and Bongard 2011)) which also explored evolving the morphologies and control policies of simulated machines in virtual

environments. These studies each had different methodologies and focuses, and the current work differs in a number of important ways.

The most visible ways in which the current study differs from all of these previous studies are (a) how morphological components are modeled and (b) the task environments within which robots evolve. In the majority of previous studies morphologies were built out of interconnected geometric primitives such as cuboids or spheres. These components are easy to model, but severely limit how complex an evolving morphology may become, and therefore restrict what task environments an evolved robot is able to succeed in. This was not a problem for the majority of earlier studies as they commonly restricted themselves to evolving locomotion over flat terrain: maximizing the distance that a robot can displace itself within a given amount of evaluation time. Here, however, more complex task environments are investigated that require the creation of more complex morphologies. Therefore, morphologies should be modeled in a manner which does not have such a low ceiling of complexity. Specifically, in the current work, morphologies are composed of a number of triangular meshes (trimeshes). Trimeshes can model arbitrary shapes and thus allow for the creation of more complex morphologies than is possible with cuboids or spheres (see Figure 8.1 for examples).

The current study also differs from much previous work in this domain in the manner by which the robots' genomes are encoded and evolved. Morphologies in the current work are encoded with Compositional Pattern Producing Network (CPPN) genomes (Stanley 2007) which are evolved using CPPN-NEAT: an extension of the widely used NeuroEvolution of Augmenting Topologies (NEAT) algorithm (Stanley and Miikkulainen 2002). CPPNs are a form of indirect encoding inspired by developmental biology possessing many advantages over other encodings (for more details see (Stanley 2007, Stanley et al. 2009, Clune et al. 2009, Clune et al. 2009, Auerbach and Bongard 2010b, Auerbach and Bongard 2010a, Auerbach and Bongard 2011)). This is particularly true for robot morphologies as it has been shown previously (Hornby and Pollack 2001, Komosinski and Rotaru-Varga 2002) that generative and developmental encodings have demonstrable benefits over direct encodings in this domain.

Following the methods introduced in (Auerbach and Bongard 2012), here robots are evolved not only to locomote over flat terrain, but to locomote in a number of more complex, icy task environments as well. However, while in that study robots were restricted to having two mechanical degrees of freedom, here robots are allowed more flexibility in their construction including the ability to utilize a greater number of degrees of freedom. How the robots evolve to use (or not use) these additional degrees of freedom in different

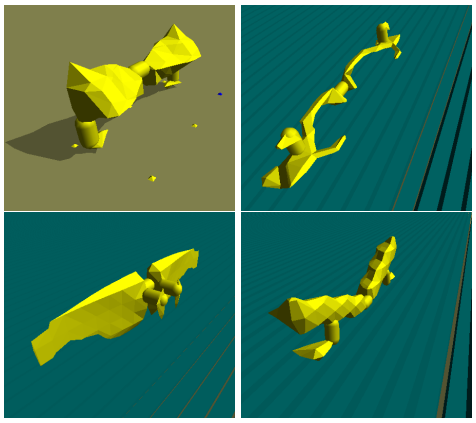


Figure 8.1: Environments and robots that evolved in them. The simple flat ground control task environment (upper left) and three of the experimental task environments with robots that evolved to locomote in each. The ground is a high friction surface, while the blue "blocks of ice" have very low friction. Videos of these robots in action are available online at `http://tinyurl.com/ALife13-Videos`

task environments is the main object of study. Here we define mechanical complexity to be the number of mechanical degrees of freedom in an evolved robot. This form of complexity can be considered an aspect of morphological complexity, but as will be shown, mechanical complexity is an orthogonal direction of complexity to the type of morphological complexity discussed in (Auerbach and Bongard 2012), and provides additional insight into the relationship between task environments and the robots evolved inside them.

The rest of this paper is laid out as follows: first the CPPN encodings are described in more detail including how they evolve and how actuated robots are produced from them. Following this the simulated task environments in which robots are evolved are described including a brief discussion of previous experiments in these task environments and why the particular task environments employed here were chosen. Next, results are presented demonstrating how the mechanical complexity of evolved robots varies across these different task environments with counterintuitive results. This is followed by a discussion of these results and what conclusions may be drawn from them.

8.2 Methods

8.2.1 CPPNs

As mentioned in the introduction this study employs Compositional Pattern Producing Networks (CPPNs) for the purpose of encoding populations of evolving robots. CPPNs may be considered a form of artificial neural network (ANN). However, while traditional ANNs are often used as control policies for evolved robots, CPPNs are more often used as genomes for producing some other object of interest. Past work has employed CPPN genomes to evolve pictures (Stanley 2007), 3D structures (Auerbach and Bongard 2010b, Clune and Lipson 2011), robot morphologies (Auerbach and Bongard 2010a, Auerbach and Bongard 2011) or traditional ANNs themselves (Stanley et al. 2009, Clune et al. 2009, Verbancsics and Stanley 2011). Here CPPNs are similarly employed to produce actuated robot morphologies.

CPPNs differ from traditional ANNs in several other important ways. While traditional ANNs typically use the same activation function (such as a sigmoid or a step function) at every node, CPPN nodes can take on one of several activation functions from a predefined set. This set typically contains functions that are symmetric such as Gaussian as well as repetitive functions such as sine or cosine. Using functions with these properties allows CPPNs to produced outputs with properties commonly seen in natural systems: symmetry, repetition, and repetition with variation. A more thorough discussion of CPPNs and their properties is beyond the scope of this paper. More details are available elsewhere in the literature ((Stanley 2007) for example).

8.2.2 Evolutionary Algorithm

Similar to most other studies employing CPPN genomes, the CPPN-NEAT (Stanley 2007) evolutionary algorithm is employed to evolve CPPNs in this work. In CPPN-NEAT the state of the art NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen 2002) algorithm for neuro-evolution is extended to evolve CPPNs. In this algorithm the CPPNs in the initial population are created to be minimally complex. That is, initially the networks do not have any internal or hidden nodes. Over evolutionary time the complexity of networks in the population is allowed to gradually increase through the creation of additional nodes and links. Often adding additional components to an evolving network will cause the fitness of its phenotype to decrease. NEAT compensates for this by dividing the population into “species” thus allowing novel

structural innovations time to mature and promoting genotypic diversity to prevent pre-mature convergence to local optima. For a complete description of how NEAT and CPPN-NEAT work, and further discussion of their beneficial properties, the reader is directed to (Stanley and Miikkulainen 2002, Stanley 2007).

8.2.3 Building Robots from CPPNs

Recently (Auerbach and Bongard 2012) we introduced a system for using CPPNs to create actuated robot morphologies composed of triangular mesh components, which is extended here. This method differs from previous studies (Auerbach and Bongard 2010a, Auerbach and Bongard 2011) where robots were constructed from evolving CPPNs by attaching spherical components to each other by means of an iterated growth procedure. While these earlier studies produced promising results, the methods they employed have several undesirable properties. The extra indirection created by the growth procedure used there prevents many of the desirable features of CPPNs (discussed above) from being realized in the morphologies they produce. Additionally, while it is easy to physically simulate spheres as they have single points of contact, it is possible to create much more complex morphologies using trimeshes.

Trimeshes do require more computational resources to simulate however, as they do not have such simple contact models as spheres, and require the use of smaller simulation step sizes to be stable in the task environments investigated here. However, all experiments described in this paper were carried out on a 7.1 teraflop supercomputing cluster[1], thus making these simulations feasible.

As opposed to employing a growth procedure to create morphologies from CPPNs the current study employs a voxel based method to create morphologies out of trimesh components. This is similar to what is done for the creation of 3D shapes in (Clune and Lipson 2011). A regular grid is placed over a region of 3D-space which defines the presence of voxel locations. In the current work this region extends from -1 to 1 (inclusive) in each dimension and grid lines are placed at intervals of 0.2. This yields a total of 11 grid lines in each dimension for a total of 1331 voxels, this is the same discretization that was applied in (Auerbach and Bongard 2012).

A candidate CPPN is iteratively queried with the (x, y, z) Cartesian coordinates at every voxel location except for the extrema in each direction. Voxel locations that exceed a predefined output threshold (0.5 in this case) are considered to contain matter, while those that do not exceed this threshold are considered to be

[1]The Vermont Advanced Computing Core (VACC), `http://www.uvm.edu/vacc`

Table 8.1: Mechanical degree of freedom parameters. This table describes the four floating point parameters evolved for each of the six potential mechanical degrees of freedom.

Parameter Name	Symbol	Range of allowed values	Interpretation
Enable Flag	f	$[0.0, 1.0]$	If $f > 0.5$, then the corresponding joint is enabled, else disabled.
Amplitude	a	$[0.25, 0.75]$	If the joint is enabled, then it is actuated by an oscillation between $-a\pi$ and $a\pi$ radians. Additionally, the joint's range of motion is restricted to this range.
Period	p	$[250.0, 1500.0]$	If the joint is enabled, then its oscillation will have a period of p simulation time steps
Phase Shift	s	$[-1.0, 1.0]$	If the joint is enabled, then its oscillation will be offset from the global oscillation by s periods.

devoid of matter. All voxels lying on one of the extrema ($|x| = 1$ or $|y| = 1$ or $|z| = 1$) are given output value 0 to ensure that the final triangular meshes have completely enclosed surfaces. Once the CPPN has been queried for every voxel location, the Marching Cubes algorithm (Lorensen and Cline 1987) is employed to create triangular meshes from the underlying voxel data. Specifically an enclosed triangular mesh is created for each connected voxel component which defines the exterior surface of a single physical shape. These triangular meshes are sent to the physics simulator where they define the exterior surfaces of solid objects and are imbued with mass. As far as the authors are aware prior to (Auerbach and Bongard 2012) physically simulating evolved, rigid body robots composed of triangular meshes had not been previously reported in the literature.

Our previous work concerned itself with investigating how different task environments affect the shapes of evolved morphologies. To accomplish this goal a single enclosed trimesh component out of the many possibly produced from a CPPN was selected and then reflected and copied in order to form a bilaterally symmetric, two mechanical degree of freedom, actuated robot. Here, however, the primary object of study is the mechanical complexity of the evolved robots, so more components are needed. The current system requires that a candidate CPPN produce at least two enclosed trimesh components. The two largest components A and B are then selected to produce an actuated robot. This is done as follows. First the vertices $a \in V(A)$

and $b \in V(B)$ are found that minimize

$$|\vec{ab}| \;\; \forall (a,b) \in V(A) \times V(B) \tag{8.1}$$

where $V(A), V(B)$ are the vertices of A, B respectively. Next, the component with larger minimum z-coordinate of A, B is translated along $\vec{ab}$ (or $\vec{ba}$) until it is 0.2 units away from the other component, and the two components are connected together via an intermediary capsule (capped cylinder) of length 0.2 units and radius 0.1 with major axis defined by $\vec{ab}$. The trimesh components may connect via this intermediary capsule by means of two joints, each being a single degree of freedom rotational (hinge) joint. These joint will have rotation normals determined by $\vec{ab}$. Specifically two rotation normals that are orthogonal to each other and orthogonal to $\vec{ab}$ are chosen. Since these two joints effectively define a universal joint, the specific normals are unimportant as long as they are orthogonal to each other and to $\vec{ab}$, so the first $\vec{n_1}$ is chosen arbitrarily (but consistently) to be orthogonal to $\vec{ab}$ and the second $\vec{n_2}$ is computed as $\vec{ab} \times \vec{n_1}$.

Once the two trimesh components are connected together with their intermediary capsule the whole object including the connecting joints is reflected across the x-axis as was done with the single trimesh component in (Auerbach and Bongard 2012). These objects are then spread apart by 0.2 units and once again connected by a capsule of this length. This capsule has its major axis along the y-axis of the coordinate system and connects the two objects at their closest points. These objects each connect to this capsule by means of hinge joints. These joints have rotation normals of $(1, 0, 0)$ and $(0, 0, -1)$ such that the joints rotate through the robot's coronal and sagitall planes respectively. Reflecting and copying the object in this manner ensures that the robots are bilaterally symmetric, which makes locomotion easier, while using two evolved trimesh components instead of the one used in prior work allows for a much greater number of morphologies and locomotion strategies. The two components within each half of the robot may connect in any orientation, and the robots may now have up to six mechanical degrees of freedom.

In addition to the trimesh producing CPPNs, each robot genome possesses a number of additional parameters that are directly encoded as was done in (Auerbach and Bongard 2012). These parameters are stored as floating point values and are used to determine aspects of the control policy as well as mechanical properties of the evolving robots. Principally, there are six parameters, one for each potential mechanical degree of freedom that act as flags for enabling or disabling a given joint. If a joint is disabled it is replaced with a

rigid connection and the remainder of the control parameters relating to that joint are ignored. However, if a joint is enabled it is actuated by means of a coupled oscillator parameterized by its amplitude, period, and phase shift from a global sinusoidal pattern generator. This results in the complete genomes being composed of a CPPN plus a 24-dimensional floating point array (four parameters for each of the six potential degrees of freedom). These floating point values are recombined and mutated in the same manner as CPPN link weights with mutation magnitudes scaled by the range of values for that parameter. Additionally, crossover on these vectors is possible in all instances of sexual reproduction since every individual contains a vector of the same dimensionality. These parameters, their ranges, and their meanings are detailed in Table 8.1. Each parameter has a mutation probability of 0.1, same as used in (Auerbach and Bongard 2012).

Allowing each degree of freedom to be enabled or disabled in this manner allows evolution to adjust the number of mechanical degrees of freedom as necessary and therefore be able to tune the mechanical complexity of the evolved robots. Moreover, encoding the control parameters in this fashion is done to keep the controllers as simple as possible so that fitness is primarily dictated by the morphologies of the robots while at the same time allowing for diverse enough behavior so that the robots can succeed in the different task environments investigated.

8.2.4 Selecting desirable robots

A candidate robot, including two enclosed triangular meshes, joint enable flags, and accompanying control parameters are sent to a physics simulator[2] and allowed to act for a fixed number of simulation time steps. Similar to (Auerbach and Bongard 2012) robots are allowed to move for $T = 12500$ time steps. While this is a much greater number of time steps than has been employed in earlier studies (e.g. 2500 in (Auerbach and Bongard 2011)) it is chosen in order to simulate a comparable amount of real world time. The reason such a large T is necessary is because a very small step size of 0.001s is used in this work. This small step size is necessary to stably simulate the sorts of simulated robots employed here in complex environments.

After the robot has completed its time in the simulator its fitness is calculated. This fitness calculation is exactly the same used in (Auerbach and Bongard 2012). It is designed to prevent evolution from "cheating" as it often does with naïve fitness functions. While a detailed explanation of the ways in which evolution may "cheat" different fitness functions is provided in that paper, here we simply state that fitness is calculated as

[2]Simulations are conducted in the Open Dynamics Engine (`http://www.ode.org`), a widely used open source, physically realistic, simulation environment.

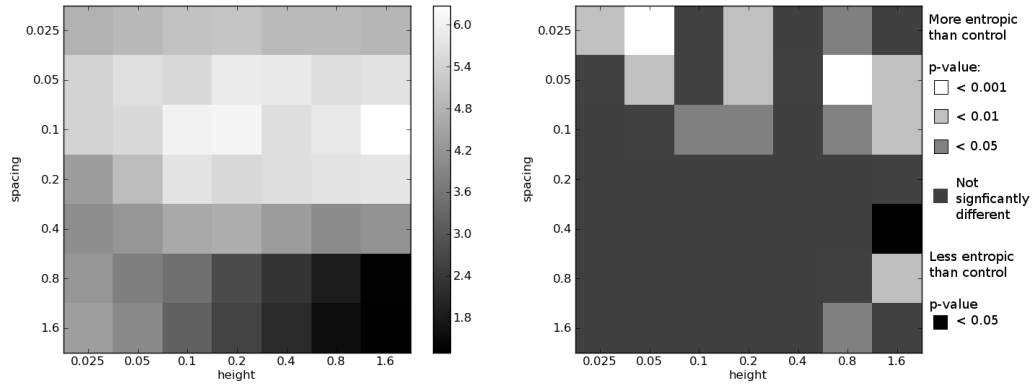


Figure 8.2: Results from Auerbach and Bongard (2012b). **(Left)** Mean distance achieved (in arbitrary ODE units) by best individual in final generation taken across 100 independent runs in each of 49 experimental task environments investigated there. For comparison the mean distance achieved from 100 independent runs in a flat ground control task environment was 5.09 units. **(Right)** The ways in which morphologies from experimental environments were more, less, or equally complex (entropic) compared to those evolved in the control task environment. The more complex experimental task environments tended to select for more complex morphologies: there were many experimental task environments where significantly more complex morphologies evolved, while only one experimental task environment selected for significantly less complex morphologies. All p-values were calculated using the Mann-Whitney U test. Figure taken from (Auerbach and Bongard 2012)

$\min p(T)_x - \max p(0)_x$ where $\min p(T)_x$ is the minimum x-coordinate of any point on the robot at time T, and $\max p(0)_x$ is the maximum x-coordinate of any point on the robot at the start of the evaluation.

Using this method of fitness evaluation robots are evolved with CPPN-NEAT for 500 generations with a population size of 150 individuals. The implementation of CPPN-NEAT including its parameter settings and CPPN activations functions are the same as employed in (Auerbach and Bongard 2012).

8.2.5 Choosing task environments

Previously, with the robots composed of a single enclosed trimesh that was reflected and copied, we explored evolving robots in a large number of task environments with the goal of studying how morphological complexity varies in relation to environmental complexity. These task environments consisted of a control environment with flat, high friction ground similar to that used in many other studies, and experimental task environments with an infinite series of low frictions rectangular solids, or "blocks of ice", fixed in place on top of the ground. These "ice blocks" were constructed such that it was impossible for a robot to gain purchase by moving over their upper surfaces but needed instead to reach into the gaps between the blocks to propel

themselves forward. This required the evolution of morphologies with appropriate physical forms. A large number of these icy task environments were explored varying according to two parameters: the height of the blocks and the spacing between the blocks. While the relative complexities of different icy environments were not considered, all the icy environments are considered to be more complex than flat ground because they have greater Kolmogorov Complexity (Kolmogorov 1965).

Figure 8.2 revisits these results. It shows, for robots evolved in that work, both how mean fitness varied across task environments and how the evolved robot morphologies differed in complexity when compared to those evolved to locomote in the flat ground, control, environment[3]. These results are employed here to select task environments for investigation with the current system.

Robots evolved with the current system, employing two trimesh components and three capped cylinders with up to six actuated mechanical degrees of freedom, are slower to simulate than those evolved previously. Due to this slowness, and additional time constraints, it was not possible to experiment with evolving robots in all 50 task environments previously investigated. In lieu of that, robots in the current study are evolved in the flat ground control environment plus five experimental environments. These five experimental environments are chosen based on previous results to be those within which robots could be successful and which selected for the most morphologically complex robots (see Figure 8.2). Specifically the five environments chosen are: blocks of ice 0.8 units tall spaced by 0.05 units (*Environment 1*), blocks of ice 0.05 units tall spaced by 0.025 units (*Environment 2*), blocks of ice 1.6 units tall spaced by 0.1 units (*Environment 3*), blocks of ice 1.6 units tall spaced by 0.05 units (*Environment 4*), and blocks of ice 0.2 units tall spaced by 0.05 units (*Environment 5*). These five task environments cover a variety of these parameters and should be a good sampling of the overall parameter space.

8.3 Results

For each of the six task environments investigated: the control plus five experimental task environments, 50 independent experimental runs of CPPN-NEAT were conducted[4]. As can be seen in Figure 8.3, in each environment studied this system is capable of evolving robots that successfully locomote in the desired direction.

[3]The measure used for comparing morphological complexities, $H_{\triangle}$, is a measure of shape complexity based on Shannon Entropy (Shannon 1948) that has been previously shown to correlate with human intuitions of complexity (Page et al. 2003, Sukumar et al. 2008). The reader is referred to (Auerbach and Bongard 2012) for a description of this measure.

[4]While 50 runs were started for each task environment, a small number of runs failed to complete for each of the experimental task environments. The results reported here only include those runs that completed successfully.

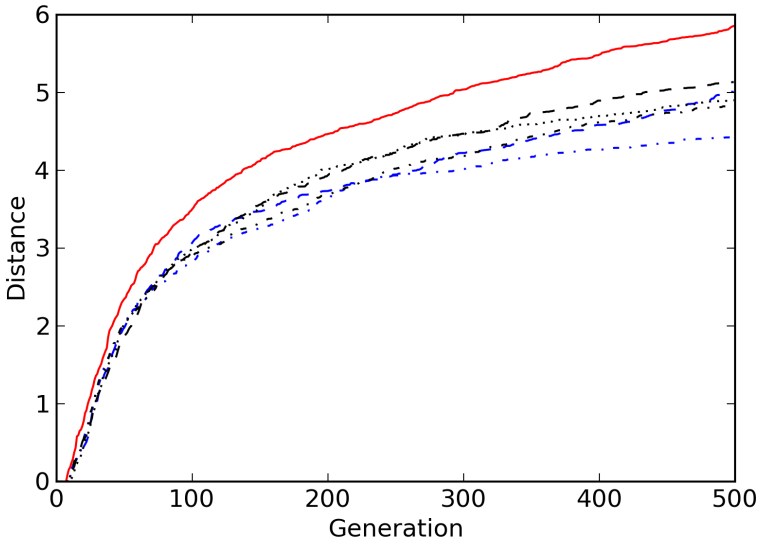


Figure 8.3: Fitness in each environment over evolutionary time. This plot shows the mean distances by generation achieved by robots evolved in the control environment (red) and each of the five experimental task environments (*env. 1* blue dashes, *env. 2* blue dash-dots, *env. 3* black dashes, *env. 4* black dash-dots, *env. 5* black dots).

Though, due to using the same number of evaluations in an enlarged search space the robots produced in the final generations here tend not to locomote as far as those evolved previously (compare to the left of Figure 8.2). However, the absolute performance of these robots is not of primary interest in this paper.

Of greater concern is how the mechanical complexity of the evolved robots varies from the simple control environment to the more complex experimental task environments. Towards this aim Figure 8.4 plots the mean number of mechanical degrees of freedom that robots evolved to use in each task environment. Counter to intuition the simple task environment actually selects for more mechanically complex robots: the robots evolved in the simple task environment have significantly more mechanical degrees of freedom on average, than those evolved in each of the five complex task environments. This is corroborated by Figure 8.5 which shows that the flat ground task environment not only selects for a greater number of mechanical degrees of freedom but that the degrees of freedom that are selected for have a significantly greater range of motion on average than the degrees of freedom in robots evolved in each of the more complex experimental task environments.

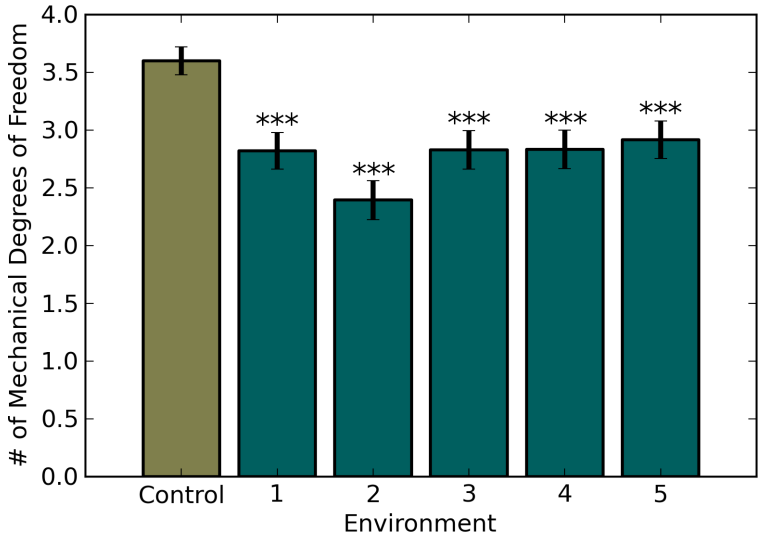


Figure 8.4: Mechanical degrees of freedom by environment. This plot depicts the mean number of mechanical degrees of freedom with standard errors for robots evolved in each task environment. Robots evolved in each of the icy task environments have significantly fewer mechanical degrees of freedom than those evolved in the control environment, p-values < 0.001 in all cases (Mann-Whitney U test).

8.4 Discussion

Why is it that the same task environments which have been shown to select for greater complexity of morphological components select for reduced mechanical complexity? Intuitively these two forms of complexity should be correlated, but this is clearly not the case here. One hypothesis is that the reduction of mechanical complexity in the icy task environments is due to them being more difficult than the flat ground task environment. As can be seen in Figure 8.3 robots are not able to evolve to locomote as far in the icy task environments as they are on flat ground. This suggests there may be fewer ways to succeed in the icy task environments, and if it is easier to succeed with less mechanical complexity than there will be selection pressure in that direction. Meanwhile, if flat ground is an easier task environment regardless of mechanical complexity there will be little selection pressure on the number of degrees of freedom of the robots evolved there. However, if this is the case, one would expect each degree of freedom of robots evolved on flat ground to be enabled or disabled with equal probability. But, from looking at Figure 8.4 it can be seen that this is clearly not the case. Robots evolved in the flat ground task environment have a significantly greater number of degrees of freedom than the three that would be expected by equal probability.

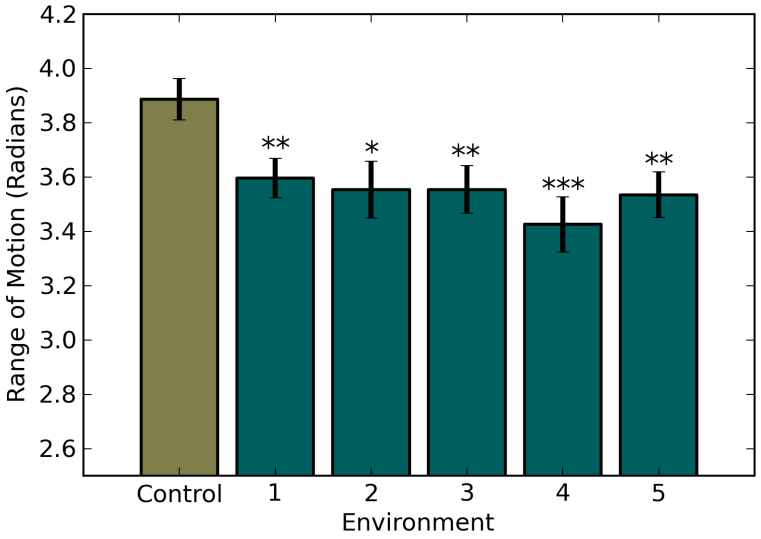


Figure 8.5: Joints' ranges of motion by environment. This plot depicts the mean range of motion in radians taken across each enabled joint (mechanical degree of freedom) with standard errors. Robots evolved in each of the icy task environments have significantly smaller ranges of motion than those evolved in the control environment. * denotes p-values < 0.05, ** denotes p-values < 0.01, and *** denotes p-values < 0.001 (Mann-Whitney U test).

Another hypothesis is that there is simply an advantage to having less mechanical complexity in the icy task environments investigated. Succeeding in these environments involves reaching into the gaps between blocks in order to gain purchase, and then coming out of the gaps in order to move forward. Since the robots evolved in this work are all driven by open loop controllers, they have no way of sensing when they are in the gaps or not. It may be that extra mechanical degrees of freedom make it more difficult for the robot to get out of its own way as it traverses the environment. In other words extra mechanical degrees of freedom driven by a sinusoidal control signal cause the robot to often catch itself in the gaps when it could be gliding forward. This seems likely to be the case. As can be seen in the video available at `http://tinyurl.com/alife13-1DOF` it is possible for robots to succeed in these task environments with only a single mechanical degree of freedom and the proper physical shape. This robot only has one actuated joint rotating horizontally but due to its shape it is able to fall into the gaps, gain purchase and glide out of them. Several such single degree of freedom robots evolved in the icy task environments, but only one such robot evolved in the control task environment and it is has substantially lower fitness.

While it is counter-intuitive that task environments that select for more complex body components select for less mechanical complexity it makes sense in this instance. It is likely, however that other task environ-

ments that are complex in different ways will select for robots that have complex body components and are more mechanically complex. For instance if there existed other obstacles in the environment that the robot needs to step over one could imagine how additional degrees of freedom would be useful in order to reach over the obstacles in order to gain purchase on their far sides in ways that would not be possible without additional degrees of freedom. Likewise if the spacing between blocks was uneven then most likely the open loop control policies employed here would be unable to succeed. If sensors and closed loop control were employed it may be advantageous to have extra degrees of freedom in order to actively sense the environment and decide how to move.

8.5 Conclusion

This work has investigated the relationship between environmental and mechanical complexity in evolved robots. Results of previous work were used to select task environments in which successful, morphologically complex, robots were previously evolved. However, counter to intuition, the robots evolved here were less mechanically complex than those evolved in a simpler control task environment. This demonstrates that these different forms of morphological complexity do not necessarily correlate with each other, but are likely orthogonal.

Moving forward it will be interesting to explore evolving robots in other task environments that are complex in different ways. It is likely that while the task environments investigated here do not select for greater mechanical complexity there exist task environments in which both greater mechanical complexity and greater complexity of body shape will be selected for. Additionally it will be of interest how control complexity varies in relation to these morphological complexity measures. To this aim the current evolutionary system will be extended to allow for more sophisticated closed loop neural network controllers. Are the task environments that select for greater morphological complexity in one way or another also those that select for greater control complexity? Or are these different forms of complexity–morphological, mechanical, and control–independent?

8.6 References

Adamatzky, A., M. Komosinski, and S. Ulatowski (2000). Software review: Framsticks. *Kybernetes: The International Journal of Systems & Cybernetics 29*(9/10), 1344–1351.

Anderson, M. (2003). Embodied Cognition: A field guide. *Artificial Intelligence 149*(1), 91–130.

Auerbach, J. E. and J. C. Bongard (2010a). Dynamic Resolution in the Co-Evolution of Morphology and Control. In *Artificial Life XII: Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems*.

Auerbach, J. E. and J. C. Bongard (2010b). Evolving CPPNs to grow three-dimensional physical structures. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Auerbach, J. E. and J. C. Bongard (2011). Evolving Complete Robots with CPPN-NEAT: The Utility of Recurrent Connections. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Auerbach, J. E. and J. C. Bongard (2012). On the relationship between environmental and morphological complexity in evolved robots. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Beer, R. D. (2008). The dynamics of brain-body-environment systems: A status report. In P. Calvo and A. Gomila (Eds.), *Handbook of Cognitive Science: An Embodied Approach*, pp. 99–120. Elsevier.

Bongard, J. and R. Pfeifer (2001). Repeated structure and dissociation of genotypic and phenotypic complexity in Artificial Ontogeny. *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO)*, 829–836.

Bongard, J. C. (2002). Evolving modular genetic regulatory networks. In *Proceedings of The IEEE 2002 Congress on Evolutionary Computation (CEC2002)*, pp. 1872–1877.

Brooks, R. (1999). *Cambrian intelligence*. MIT Press Cambridge, Mass.

Clune, J., B. Beckmann, C. Ofria, and R. Pennock (2009). Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computing*, pp. 2764–2771.

Clune, J. and H. Lipson (2011). Evolving 3D objects with a generative encoding inspired by developmental biology. In *Proceedings of the Eleventh European Conference on Artificial Life (ECAL)*, pp. 144–148.

Clune, J., R. T. Pennock, and C. Ofria (2009). The sensitivity of HyperNEAT to different geometric representations of a problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. *Procs. of the Fourth European Conf. on Artificial Life*, 205–213.

Harvey, I., P. Husbands, D. Cliff, A. Thompson, and N. Jakobi (1997). Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems 20*, 205–224.

Hornby, G. S. and J. B. Pollack (2001). Body-brain co-evolution using L-systems as a generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, pp. 868–875. Morgan Kaufmann.

Kolmogorov, A. N. (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission 1*(1), 1–7.

Komosinski, M. and A. Rotaru-Varga (2002). Comparison of different genotype encodings for simulated three-dimensional agents. *Artif. Life 7*(4), 395–418.

Lipson, H. and J. Pollack (2000). Automatic design and manufacture of robotic lifeforms. *Nature 406*, 974–978.

Lorensen, W. E. and H. E. Cline (1987). Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 163–169.

Lund, H. H., J. Hallam, and W. Lee (1997). Evolving robot morphology. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*, Piscataway, NJ, USA. IEEE Press.

Mautner, C. and R. Belew (2000). Evolving robot morphology and control. *Artificial Life and Robotics 4*(3), 130–136.

Nolfi, S. and D. Floreano (2000). *Evolutionary Robotics: The Biology,Intelligence,and Technology*. Cambridge, MA, USA: MIT Press.

Page, D., A. Koschan, S. Sukumar, B. Roui-Abidi, and M. Abidi (2003). Shape analysis algorithm based on information theory. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, Volume 1, pp. I – 229–32 vol.1.

Paul, C. (2006). Morphological computation: A basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems 54*(8), 619–630.

Pfeifer, R. and J. Bongard (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal 27*.

Sims, K. (1994). Evolving 3D morphology and behavior by competition. *Artif. Life 1*(4), 353–372.

Stanley, K., D. D'Ambrosio, and J. Gauci (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life 15*(2), 185–212.

Stanley, K. and R. Miikkulainen (2003). A taxonomy for artificial embryogeny. *Artificial Life 9*(2), 93–130.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines 8*(2), 131–162.

Stanley, K. O. and R. Miikkulainen (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation 10*(2), 99–127.

Sukumar, S., D. Page, A. Koschan, and M. Abidi (2008). Towards understanding what makes 3d objects appear simple or complex. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2008, Sixth IEEE Workshop on Perceptual Organization in Computer Vision (POCV)*.

Verbancsics, P. and K. O. Stanley (2011). Constraining connectivity to encourage modularity in HyperNEAT. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Chapter 9

Environmental Influence on the Evolution of Morphological Complexity in Machines

Whether, when, how, and why increased complexity evolves in biological populations is a long-standing open question. Here we demonstrate, with a population of evolving robots, and an information-theoretic metric of morphological complexity, that morphologies having greater complexity evolve in complex environments when compared to simple environments. Moreover, these complex environments select for increased morphological complexity, while simple environments select against the complexifying bias inherent in the evolutionary algorithm employed.

9.1 Introduction

The "arrow of complexity" hypothesis (Bedau et al. 1998) posits that the products of evolutionary systems tend to increase in complexity over time. Whether such a tendency exists is a long standing open question (McShea 1991, McShea 1996, Feldman and Crutchfield 1998, Adami 2002, Miconi 2008). While it

seems evident that more complex organisms exist today than at the advent of life, simple (single-celled) organisms continue to persist in large numbers, so it is clear that evolution does not guarantee complexity must increase over time. Moreover, loss of complexity has been observed in many species (McCoy 1977, Jeffery and Martasian 1998, Gould 1996). This begs the question: under what circumstances will complexity increase or decrease over evolutionary time? It is likely that particular environmental conditions are more likely to select for increased complexity than others.

As argued by proponents of embodied cognition, intelligent behavior emerges from the interplay between an organism's brain, body and environment (Brooks 1999, Pfeifer and Scheier 1999, Anderson 2003, Pfeifer and Bongard 2006, Beer 2008). Therefore, if the ecological niche of a species remains constant and its body plan is evolutionarily constrained, then the neural system must adapt in order to succeed under this particular set of circumstances. This may be investigated experimentally through the use of evolving robots (Harvey et al. 1997, Nolfi and Floreano 2000) which stand in for biological organisms. For instance, it has been demonstrated (Pfeifer and Scheier 1999, Paul 2006) that the complexity of an evolved neural system depends on the particular morphology it is controlling: in a given task environment certain morphologies can readily succeed with simple controllers, while other morphologies require the discovery of much more complex neural systems, or may prevent success altogether.

Another corollary of embodied cognition is that different environments will impose different selection pressures on the brains and bodies of organisms evolving in them. This can be studied by observing how organisms evolve in different environments. For instance, Passy (2002) demonstrated that the morphological complexity of benthic colonial diatoms (measured as their fractal dimension) is significantly correlated with the variability of the environmental niches in which they are found. However, the biological evidence for a correlation between environmental and morphological complexity is sparse. This is in part because it is difficult to isolate systems where this may be studied effectively. Ideally, it would be desirable to perform controlled investigations where environmental complexity is under experimental control. While it may be possible to perform such investigations directly in biological organisms given enough time and resources, through the evolution of virtual organisms (Sims 1994a) in physically realistic simulation environments, it becomes possible to perform evolutionary experiments *in silico* with much greater speed and more precise control over experimental conditions.

Using *in silico* evolution to act on both the morphologies and nervous systems of simulated organisms or robots was first demonstrated by Sims (Sims 1994a), and has since been followed by a number of other studies (e.g. (Lund et al. 1997, Adamatzky et al. 2000, Mautner and Belew 2000, Lipson and Pollack 2000, Hornby and Pollack 2001, Komosinski and Rotaru-Varga 2002, Stanley and Miikkulainen 2003, Eggenberger 1997, Bongard and Pfeifer 2001, Bongard 2002b, Auerbach and Bongard 2010a, Auerbach and Bongard 2011)). These studies employ a variety of experimental techniques including different genetic encodings, morphological systems (such as branching structures or cellular aggregations), and evolutionary models. However, by constructing morphologies out of a relatively small number of geometric primitives, all of these studies were severely limited in the complexity of the morphologies which they could evolve, and therefore do not offer good test beds for investigating how different environments select for different degrees of morphological complexity.

Recently, we introduced a new method for evolving complete robots that is capable of producing a much greater diversity of morphologies (Auerbach and Bongard 2012b). By using it to evolve robots with restricted control systems in a variety of environments it was possible to demonstrate that more complex environments tend to select for more complex morphologies, as the principles of embodied cognition would suggest. However, it is unclear how the different environments impose selection pressures on the complexity of the evolving morphologies. Here, the results of (Auerbach and Bongard 2012b) are extended to demonstrate that complex environments select for increased morphological complexity, while simple environments select against the complexifying bias inherent in the evolutionary algorithm employed. This result is corroborated by additional experiments employing a multi-objective selection mechanism to select for simplicity in addition to behavioral competency. This filters out morphological complexity that arises due to biases in the underlying algorithm or genetic drift, and only allows complexity that confers a selective advantage on the simulated organism to remain. Under this regime, the differences between the morphological complexities of robots evolved in simple and complex environments become even more pronounced, further supporting the active role of environment in determining morphological complexity.

The remainder of this paper is organized as follows: first the methods used for evolving robots are presented. This includes a discussion of the advantages offered over previous systems, a description of the genotypic encoding employed and how this encoding is translated into an autonomous robot, a presentation of the algorithms used to evolve these robots, and finally a characterization of the simulated environments

within which the robots evolve. Next, results are presented along with a discussion of how morphological complexity may be calculated using geometric properties and information theoretic measures. These techniques are then applied to analyze how different environments influence the ways in which the environment influences the evolution of morphological complexity including comparisons with neutral shadow models that are free from environmental factors. After this, it is shown how these results are corroborated through the use of multi-objective selection which evolves robots not only for competency in their given task environment, but also for their morphological simplicity. The last section then presents conclusions and some discussion of how the techniques presented in this paper may be applied in future work.

9.2 Methods

This paper employs a new method for evolving the morphologies and neural systems of simulated robots. Robots are evolved in a variety of environments in order to better understand the role of environment in shaping the complexity of evolving robots. While it draws inspiration from the above mentioned studies in which the morphologies and controllers of robots were also evolved, the system presented here has several advantages which make it better suited for studying the evolution of morphological complexity.

The first advantage relates to the task environments within which robots evolve. The majority of the studies mentioned above were restricted to evolving robots to locomote over flat terrain. While investigating this task has yielded interesting results, it suffers from its simplicity: simple morphologies composed of just a few cuboids or spheres are all that are needed to be successful. Even when more challenging task environments have been explored (e.g. those investigated in (Lassabe et al. 2007)), they employed morphologies composed of a small collection of cuboids and therefore the maximum complexity of their evolved morphologies was severely limited. In the current work, a variety of task environments with interesting properties are investigated, and morphologies with greater geometric detail are used, so it is possible to study how environment influences the evolution of morphological complexity.

Another advantage of the current system is the genetic encoding employed. As has been demonstrated in the past (Hornby and Pollack 2001, Komosinski and Rotaru-Varga 2002), generative and developmental encodings offer demonstrable benefits over direct encodings for evolving robot morphologies. Accordingly, the morphologies in this study are created from a specific generative encoding: Compositional Pattern Produc-

ing Networks (CPPNs) (Stanley 2007). CPPNs have been shown to possess a host of advantages over other encodings. From this encoding it is possible to construct morphologies out of triangular meshes (trimeshes), thus allowing for the creation of a much greater diversity of morphologies than is possible with geometric primitives (see Figs. 9.1 and 9.4 for examples of morphologies evolved with the current system). Moreover, the CPPN genomes which encode the morphologies are evolved using CPPN-NEAT: an extension of the widely used NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen 2002) algorithm, which presents many advantages over previous approaches for evolving robot morphologies.

The use of these techniques is made possible by the vast amount of computational resources that many modern researchers have access to. These resources are necessary to run large numbers of physical simulations at small enough step sizes to produce physically plausible results. All the experiments presented in this paper are carried out on a 7.1 teraflop supercomputing cluster. Without access to such a distributed computing system one single evolutionary trial from one single experiment would take multiple days to complete on a standard personal computer. But, when using the cluster, an entire experiment (of 100 trials) can be conducted in less than one day thus allowing for experimentation with a large number of environments, and conducting enough experimental trials to perform meaningful statistical analyses.

These techniques are described in more detail below, starting with a description of the CPPN genomes and the algorithms used to evolve them.

9.2.1 CPPNs

CPPNs (Stanley 2007) are a form of artificial neural network (ANN) which differ from traditional ANNs in several ways. While each internal node in a traditional ANN typically has the same activation function (such as a sigmoid or a step function), CPPN nodes can take on one of several activation functions from a predefined set. This function set often includes functions that are repetitive, such as sine or cosine, as well as symmetric functions, such as Gaussian, thus allowing for motifs seen in natural systems: symmetry, repetition, and repetition with variation. Additionally, CPPNs are often used as generative systems to produce some other object of interest, such as images (Secretan et al. 2011), 3-D structures (Auerbach and Bongard 2010b, Clune and Lipson 2011), robot morphologies (Auerbach and Bongard 2010a, Auerbach and Bongard 2011) or traditional ANNs themselves(Stanley et al. 2009, Gauci and Stanley 2008, Clune et al. 2009, Gauci and Stanley 2010, Lee et al. 2013). This is in contrast to the typical, direct application of ANNs as robot

control architectures or classifiers. Here CPPNs are used as such a generative encoding to produce actuated robot body plans.

CPPNs act as functions of geometry. Geometric coordinates meaningful to the object being represented are fed as inputs to the CPPN. These input values are passed through the various connections of the CPPN from node to node. Each node aggregates its inputs by taking a weighted sum of the values output by each upstream node (weights are specific to each connection) and outputs the result of applying a particular activation function (specific to that node) to this weighted sum. By passing the inputs through subsequent nodes the activation functions are composed to produce novel outputs while maintaining features of the different functions (hence the "compositional" aspect of CPPNs). Additionally, since these functions are chosen to have desirable properties present across a wide range of natural systems, as discussed above, CPPNs are capable of directly producing structures which in nature require the iteration of a developmental process. For a more in depth description of CPPNs, and further discussion on their ability to act as an abstraction of development, the reader is referred to (Stanley 2007).

9.2.2 Evolutionary Algorithm

In this study CPPNs are evolved via CPPN-NEAT (Stanley 2007). CPPN-NEAT is an extension of the NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen 2002) method of neuro-evolution. NEAT is capable of evolving not only connection weights for existing network topologies, but also the network topologies themselves. Its operation is based on a few key ideas. First, the initial population is comprised of minimal networks (those without any internal or hidden nodes), which may then gradually increase in complexity over evolutionary time through structural mutations which add new nodes and links to the network. When a new node or link is created in this manner it is assigned a unique historical marking. These historical markings are inherited during reproduction and allow meaningful crossovers to occur without the use of expensive graph matching procedures. Additionally, these markings are used to divide the population into "species" of similar network topologies. Speciation promotes genotypic diversity and, because competition is primarily intraspecies, novel structural innovations are given time to mature before directly competing with individuals in other species.

CPPN-NEAT extends NEAT to evolve CPPNs. Effectively, this means that since nodes are no longer restricted to having sigmoid activation functions, each node contains an additional field which specifies its

own activation function. When a new node is added to a network it is assigned a random function from a predefined set. Additionally, the compatibility distance metric used for speciation is modified to incorporate the number of different activation functions between two networks. In all other respects, CPPN-NEAT behaves the same as NEAT.

NEAT and CPPN-NEAT have successfully evolved ANNs and CPPNs for a variety tasks (Stanley and Miikkulainen 2002, Stanley et al. 2005, Gauci and Stanley 2008, Clune et al. 2009, Secretan et al. 2011, Clune and Lipson 2011) which makes CPPN-NEAT a good option for evolving the CPPNs used in this study. Moreover, CPPN-NEAT's ability to systematically increase network complexity over evolutionary time as needed should lend itself well to studying how morphologies increase in complexity when evolving inside different environments. For a more thorough description of these algorithms, including additional details of the mechanisms discussed above, please refer to (Stanley and Miikkulainen 2002, Stanley 2007).

9.2.3 Building Robots from CPPNs

In previous studies (Auerbach and Bongard 2010a, Auerbach and Bongard 2011), robots were constructed out of spherical components from evolving CPPNs by means of an iterated growth procedure. This procedure involved starting at a specific initial point and attaching spheres to grow outwards by means of querying the CPPN genome locally and placing newly created spheres in a priority queue whereby they could be selected as attachment points for additional spheres. This process would repeat until a complete robot was grown.

While promising results were produced by the system presented in those papers, it has several drawbacks. As described above, CPPNs are capable of replicating the products of development without having to simulate developmental change over time. In fact, the additional indirection added by this growth procedure often prevents desirable features of the CPPNs' outputs – such as symmetry and repetition – from being realized in the resulting morphologies. Moreover, while spheres are easy to physically simulate due to their single points of contact, and therefore morphologies with a small number of spheres can be cheaply simulated, the computational costs become too large when trying to model sufficiently complex physical shapes with spheres. This is due to the creation of large kinematic chains, across which various forces must be propagated. Because of these considerations, an alternative method is employed in this work.

In lieu of the growth procedure just described, the current study employs a voxel based method to create morphological components out of triangular meshes (trimeshes) similar to what is done for the creation of

3-D shapes in (Clune and Lipson 2011). A regular grid is placed over a region of 3-D space which defines the presence of voxel locations. In the current work this region extends from -1 to 1 (inclusive) in each dimension and grid lines are placed at intervals of 0.2. This yields a total of 11 grid lines in each dimension for a total of 1331 voxels.

A candidate CPPN is iteratively queried with the (x, y, z) Cartesian coordinates at every voxel location except for the extrema in each direction. Querying a CPPN at a given location involves resetting all node values, and updating the CPPN for a fixed number of iterations (in this case 10) before the output value is retrieved. This procedure is employed in order to extract consistent output signals from networks with recurrent connections, which may fall into cyclic or chaotic attractors. Previously (Auerbach and Bongard 2011), it was found that allowing recurrent connections in robot-generating CPPNs increased their evolvability. Voxel locations that exceed a predefined output threshold (0.5 in this case) are considered to contain matter, while those that do not exceed this threshold are considered to be devoid of matter. All voxels lying on one of the extrema ($|x| = 1$ or $|y| = 1$ or $|z| = 1$) are given output value 0 to ensure that the final triangular meshes have completely enclosed surfaces. Once the CPPN has been queried for every voxel location, the Marching Cubes algorithm (Lorensen and Cline 1987) is employed to create triangular meshes from the underlying voxel data. Specifically, an enclosed triangular mesh is created for each connected voxel component which defines the exterior surface of a single physical shape. These triangular meshes are then sent to the physics simulator where they define the exterior surface of a solid object and are imbued with mass. This is the first instance of physically simulating evolved, rigid body robots composed of triangular meshes.

Since the purpose of this study is to investigate how different task environments affect the shapes of evolved morphologies, a number of simplifications are used in order to concentrate on the physical shapes of the evolved robots and control for other factors that may influence their performance. From the multiple enclosed trimesh components that could be produced when querying a single CPPN only one of these (the largest in terms of number of triangles) is used in the resulting robot. This single component is copied and reflected across the x-axis. The resulting components (the original and its mirror image) are then spread apart by 0.2 units and a capsule of this length is placed between them such that it connects their two closest points. The two trimesh components each connect to this capsule by means of a hinge joint. These joints have rotation normals of $(1, 0, 0)$ and $(0, 0, -1)$ such that the joints rotate through the robot's coronal and sagittal planes respectively. Reflecting and copying a single component like this ensures that all robots have

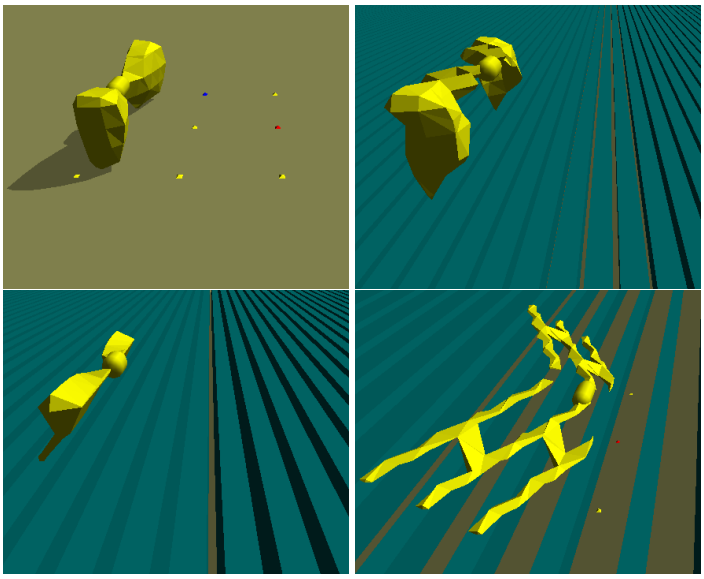


Figure 9.1: Evolved robots and their environments. The control environment (top left) and three experimental environments are shown with robots that evolved to locomote successfully in each. The ground is a high friction surface, while the blue "blocks of ice" have very low friction. To view videos of these robots visit `http://www.cs.uvm.edu/~jauerbac`

the same degrees of freedom and ensures that the robots are all bilaterally symmetric (which should facilitate locomotion) while at the same time it allows for a very large number of different morphologies due to the flexibility of the CPPN representation and trimesh model.

The two degrees of freedom of each robot are actuated by means of coupled oscillators. Each of the two oscillators is parameterized by several parameters: amplitude, period, and phase shift. These six parameters (three parameters apiece for each of the two joints) are directly encoded in the genome of the evolving robots as floating point values so that the genome is in actuality a CPPN plus a six dimensional floating point array. These floating point values are recombined and mutated in the same manner as CPPN link weights with mutation magnitudes scaled by the range of values for that parameter[1]. Additionally, crossover on these vectors is possible in all instances of sexual reproduction since every individual contains a vector of the same dimensionality. Values for these parameters are constrained to predefined ranges: amplitude, $a \in \left[\frac{\pi}{4}, \frac{3\pi}{4}\right]$ radians (so that the hinge rotates between $-a$ and a radians), period $\in [250, 1500]$ simulation time steps (or

[1]Note that the original results presented in (Auerbach and Bongard 2012b) were from experiments that did not have their mutation magnitudes scaled in this manner, but rather employed the same mutation magnitudes as used for link weights. All results reported here are based on experiments in which these mutation magnitudes have been appropriately scaled. This change was made because it was found that scaling link weights in this manner consistently evolved robots that achieved higher fitness than those evolved without this scaling. However, the conclusions of that paper still hold, as is discussed below.

equivalently $[2, 12]\%$ of the total evaluation time) and phase shift $\in [-1, 1]$ periods. Each parameter has a mutation probability of 0.1, which was chosen experimentally.

Encoding the control parameters in this fashion is done to keep the controllers as simple as possible so that fitness is primarily dictated by the physical form of the robots, while at the same time allowing for diverse enough behavior so that the robots can succeed in the different task environments.

9.2.4 Selecting desirable robots

The focus of this study is on how varying environmental complexity changes the evolutionary pressures that are imposed on the evolving robot morphologies. Towards this aim a simple task is chosen which can be accomplished with more or less difficulty in a variety of environments. Specifically, as in previous work (e.g. (Sims 1994b, Lipson and Pollack 2000, Hornby and Pollack 2001, Bongard 2002a, Auerbach and Bongard 2011)), the task investigated here is to maximize directed displacement in a fixed amount of time, though this is done across a range of environments and not just on flat ground.

A candidate robot morphology (triangular mesh) and accompanying control parameters are sent to a physics simulator[2] and allowed to act for a fixed number of simulation time steps. Since trimeshes can be arbitrarily shaped and, unlike spheres, may simultaneously contact the environment at several points, it is necessary to use a much smaller step size than has been used in previous work in order to get physically realistic behavior. Specifically, a step size of 0.001s is used in this work. Because of this smaller step size a proportionally larger number of time steps are needed to achieve the same effective simulation length. Here robots are evaluated for $T = 12500$ time steps.

Single objective selection

After the robot has completed its time in the simulator its fitness is calculated. How exactly this fitness is calculated takes some care, because evolution often finds ways to "cheat" naïve fitness functions, especially when the task environment is difficult. For example, if fitness only considers the positions of the robot's center of mass, C, and takes fitness as $C(T)_x - C(0)_x$ where $C(t)_x$ is the x-coordinate of the robot's center of mass at time t and T is the simulation length, then in environments where locomotion is difficult evolution will tend to find solutions where C is initially raised far off the ground so that its displacement can be maximized

[2]Simulations are conducted in the Open Dynamics Engine (`http://www.ode.org`), a widely used open source, physically realistic simulation environment.

by falling forward. This is a local optimum in this fitness landscape. Similarly, if one tries to eliminate this cheating by only considering the trailing point of the robot so that fitness is $\min p(T)_x - \min p(0)_x$ where $\min p(t)_x$ is the smallest x-coordinate across all points on the robot at time t, falling forward can still be an effective solution (and is still a local optimum) in difficult environments if morphologies are created which have posterior protrusions and thus make $\min p(0)_x$ as small as possible.

In light of these considerations, the fitness function employed in all environments in this study is

$$f_1 = \min p(T)_x - \max p(0)_x \tag{9.1}$$

where all coordinates are taken in terms of ODE units, which may be thought of as meters.

With this fitness function, falling forward will not be rewarded, because the maximum fitness that can be achieved by pivoting about a single point will be 0, and so a robot must actually displace its whole body forward to be rewarded.

Multi-Objective Selection

In the initial set of experiments described below f_1 is the only fitness function employed, but in a second set of experiments f_1 is used in conjunction with a minimal complexity fitness function through the use of multi-objective selection(Deb 2001, Fonseca and Fleming 1993). By selecting for robots that are morphologically simple and are able to displace as far as possible in their given environment, it should be possible to evolve robots that are no more complex than they need to be in order to succeed.

In order to evolve CPPNs using multi-objective selection it is necessary to modify CPPN-NEAT to use multiple fitness criteria. In lieu of the speciation and selection mechanisms employed by CPPN-NEAT, the widely used Non-dominated Sorting Genetic Algorithm-II (NSGA-II) (Deb et al. 2002) is used for selection. The two primary fitness functions to be maximized by NSGA-II are f_1 (see Eqn. 9.1) and f_2: a term which selects for morphological simplicity and will be defined below (see Eqn. 9.4). Additionally, preliminary experiments determined that including a genotypic diversity objective based on NEAT's compatibility metric consistently improved performance on both primary objectives, and so this additional objective is included in all reported experiments. It is thought that this term is useful because, like NEAT's speciation mechanism, it provides a means for solutions in different parts of the genotype space to evolve without competing directly

with one another[3]. Specifically, the genotypic diversity measure employed here (to be maximized) for each individual is calculated as the sum of the compatibility distances to its k nearest neighbors. A value of $k = 15$ is employed in all experiments, because it was found to achieve the best performance during preliminary experimentation.

9.2.5 Exploring environments

As mentioned previously, the goal of this study is to investigate how varying the complexity of task environments imposes different selection pressures on the complexity of the evolved robots. To accomplish this goal, robots are evolved in a range of environments with tunable parameters that can effectively increase or decrease the difficulty of the task. For each environment investigated, 100 independent evolutionary trials of CPPN-NEAT are run for 500 generations with a population size of 150 in the single objective case and another 100 independent evolutionary trials of NSGA-II are run for 500 generations also with a population size of 150 in the multi-objective case. In each of these experiments the CPPN internal nodes are allowed to use the signed cosine, Gaussian, and sigmoid activation functions to allow for repetition, symmetry and variation. Additionally, as mentioned above, recurrent CPPN connections are allowed. All other parameters of CPPN-NEAT are kept at the default values provided with the C++ implementation of HyperNEAT[4] except for the addition of the floating point array encoding the control parameters, and an improved selection mechanism[5]. For a complete list of algorithm parameters, see Appendix A.

The first environment in which robots are evolved is composed only of flat, high friction ground similar to previous work. The robots evolved in this simple environment are considered control cases to compare with robots evolved in other environments. Subsequent environments are more complex: they all consist of an infinite series of low friction rectangular solids ("blocks of ice") over which a robot must locomote[6]. These "ice blocks" are constructed such that it is impossible for a robot to gain purchase by moving over their upper surfaces, but must instead reach into the gaps between the blocks to propel themselves forward. This requires

[3] As detailed in (Mouret and Doncieux 2012) it is likely that performance could be improved even further by including a behavioral diversity metric, but it is not clear what an appropriate behavioral diversity metric is when evolving morphologies, while the NEAT compatibility metric is readily available.

[4] Available at `http://eplex.cs.ucf.edu/hyperNEATpage/HyperNEAT.html`

[5] The authors were made aware through personal correspondence of a bug in the selection mechanism in previous versions of the HyperNEAT C++ distribution. The code was patched to fix this bug (and thus behave as described in the literature) before the current experiments were run.

[6] These environments are considered more complex than flat ground, because they have a greater description length or Kolmogorov Complexity (Kolmogorov 1965).

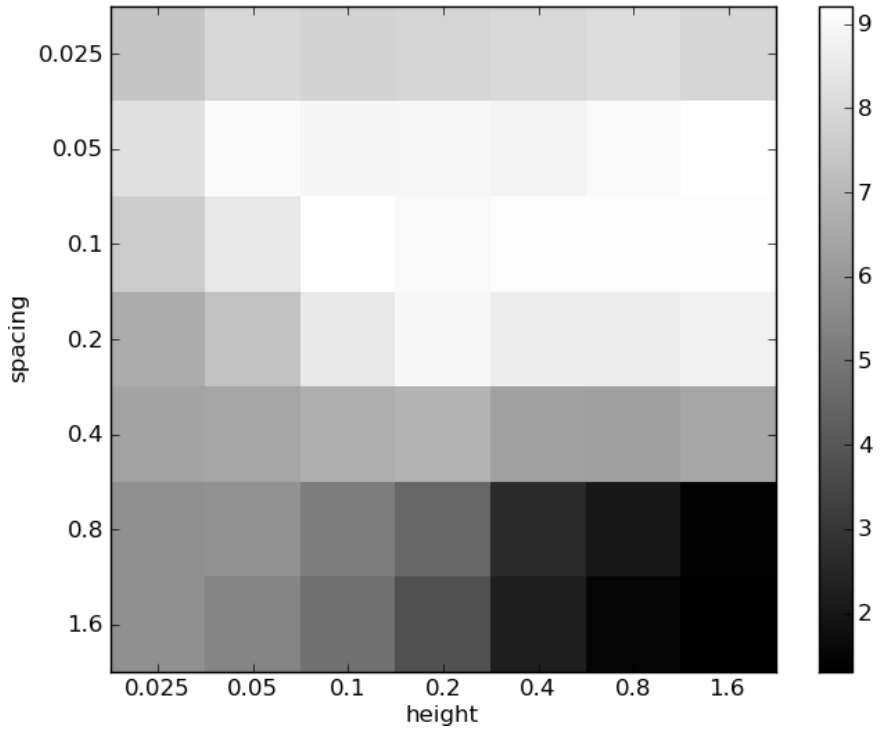


Figure 9.2: Mean distance achieved in each environment. This plot shows the mean distance achieved by the final generation champion taken across the 100 independent trials of CPPN-NEAT in each of the 49 experimental environments. For comparison the mean distance achieved from the 100 independent trials in the control environment is 7.11 units.

the evolution of morphologies with appropriate physical forms. These "icy" environments vary according to two parameters: the height of the blocks and the spacing between the blocks. Each of these parameters varies from 0.025 units to 1.6 units exponentially for a total of $7 * 7 = 49$ different environments. The exponential scaling is used in order to cover a range of parameters which produce qualitatively different environments. Fig. 9.1 shows a sampling of these environments and robots that evolve inside them.

9.3 Results

After completing 100 trials of CPPN-NEAT in the control environment and another 100 trials of CPPN-NEAT for each of the 49 experimental environments (for a total of $50 * 100 = 5000$ evolutionary trials) the most first question is to consider is: how difficult are these different experimental environments? Or, put another way, how successful is this evolutionary system at producing locomoting robots in each of these environments?

Fig. 9.2 shows the mean distance that the best individuals from each CPPN-NEAT trial are able to locomote (taken across the 100 independent trials) in each experimental environment. This figure demonstrates that there is a clear relationship between these environmental parameters and the difficulty of the task. Specifically, starting in the lower right of this matrix where both the spacing and the height of blocks are large the task becomes very difficult. The robots all become stuck in the gaps between blocks and are unable to successfully locomote. Keeping the spacing constant and decreasing the block height gradually makes the task easier. The robots are able to navigate over these smaller blocks, and displace themselves at least several body lengths. Once the height has been reduced to 0.025 units the blocks are so small that the environment becomes very similar to flat ground, and in fact distances achieved by robots in the lower left environments are not significantly different from those of the control environment.

As the spacing between the blocks is reduced the robots are no longer able to behave as they would on flat ground, but instead must find ways to move along the tops of the blocks while finding a means of gaining purchase by reaching into the gaps. The height of the blocks loses importance in this part of the parameter space but still has an effect (though opposite to when the spacing is large). Here the general pattern is for taller blocks to make the task easier: at these spacings, when the blocks are short, the gaps are small and the ways in which robots can gain purchase are limited, but when the blocks are taller the gaps have greater volume, and therefore a multitude of different shapes can be successful. Finally at the top of the matrix, when the spacing is smallest, block height ceases to have an impact because however narrow a robot's components are it can only reach a small distance into the gaps.

For a better understanding of how the evolved robots behave in each of these environments it is helpful to observe their behavior. For this purpose, videos of robots evolved in each environment are available on the web at `http://www.cs.uvm.edu/~jauerbac`.

9.4 Discussion

It is clear that different environments in this parameterization present the evolutionary system with varying degrees of difficulty, but do they also select for different sorts of morphologies? And if so, can these differences be quantified?

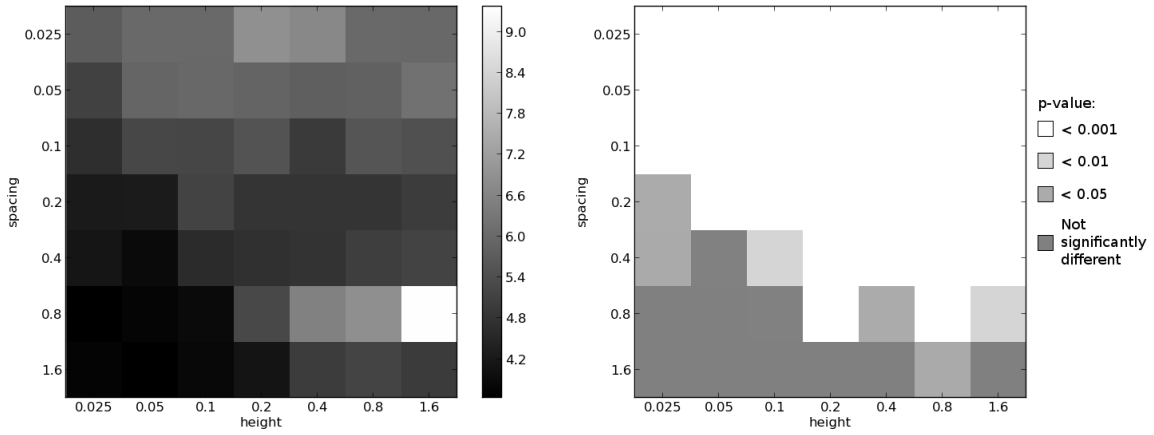


Figure 9.3: How space filling the evolved morphologies are. *Left*: Mean ratio between the volume of the robot morphology's Axis Aligned Bounding Box (AABB) and the volume of the morphology itself for each of the experimental environments. The best robots from each trial of the control experiment have a mean of 3.84 for this ratio, similar to the black squares in this plot. *Right*: Significance of the difference of this ratio in each experimental environment compared to the control environment. The ratio is significantly greater (morphologies are significantly less space filling) on average in the majority of experimental environments. There are no experimental environments in which this ratio is significantly smaller than that of the control. All p-values calculated using the Mann-Whitney U test.
N.B. while the robots evolved in some environments such as (spacing 0.8, height 1.6) have very high means, they also have very large variances and so may not be as different from the control environment as would be expected from considering their means alone.

One simple way to study this question is to consider how space filling the evolved morphologies are. This can be done by computing the ratio of the volume of a morphology's Axis Aligned Bounding Box (AABB) to the volume of that morphology itself[7].

Fig. 9.3 shows the mean values of this ratio, once again taken across the 100 best of trial individuals from each experimental environment. Also plotted is how significantly different this ratio is, on average, in each experimental environment when compared to the best of trial individuals from the control environment. In the majority of experimental environments this ratio is significantly greater from that of the robots in the control experiment. This demonstrates that these environments do in fact influence the morphologies of the robots which evolve inside them in quantifiable ways: they are space filling than those evolved in the control environment for a large portion of the parameter space. Additionally, Fig. 9.3 (left) shows how (at least) one aspect of how morphology gradually changes as one moves through this environmental parameter space: the evolved morphologies become less space filling as height increases and spacing decreases. This lends support

[7]For simplicity all morphological measures are computed on the single enclosed trimesh object that is produced by Marching Cubes for a CPPN, i.e. the reflected copy of this trimesh and the connecting capsule are not considered.

to the chosen parameterization being a good one for the purpose of studying how the morphologies of robots are affected by the environment in which they evolve.

It is clear that the morphologies which evolve in these environments vary in quantifiable (and significant) ways across this parameter space. The question now becomes: do some or all of these environments actually select for more complex morphologies than those that evolve to locomote over flat ground?

There are many ways one might think to quantify the complexity of an evolved morphology. Different measures of how space-filling a morphology is, such as the AABB ratio presented above, or its surface area to volume ratio, or measures of how concave a morphology is (e.g. the ratio of a morphology's volume to that of the convex hull of its points) may all hint at how complex a morphology is. However, each of these measures may be deceived by relatively simple body shapes, such as those that are very flat, or contain large simple concavities (e.g. a C shape).

9.4.1 Entropy of curvature

Instead, it is useful to think about the complexity of a body shape in information theoretic terms. One commonly used measure of complexity is Shannon's Entropy (Shannon 1948), which measures the information content of a random variable. Recent work (Page et al. 2003, Sukumar et al. 2008) has demonstrated how Shannon Entropy can be applied to measuring the complexity of a 3-D object by considering the curvature of the object as a random variable. In fact, quantifying the complexity of 3-D objects in this way has been shown to strongly correlate with human observers' notions of complexity (Sukumar et al. 2008). In the space below, the building blocks of computing this measure are presented. The reader is referred to (Do Carmo 1976, Page et al. 2003, Surazhsky et al. 2003, Sukumar et al. 2008) for more in-depth discussions of their theoretical underpinnings.

Given a random variable x with a probability density function (PDF) $p(x)$, entropy H is defined as

$$H = -\sum_i p_i \log p_i \tag{9.2}$$

where $p(x)$ is discretized such that $p_i = \int_{x_{i-1}}^{x_i} p(x)\,\mathrm{d}x$ where the x_is are specific values of x.

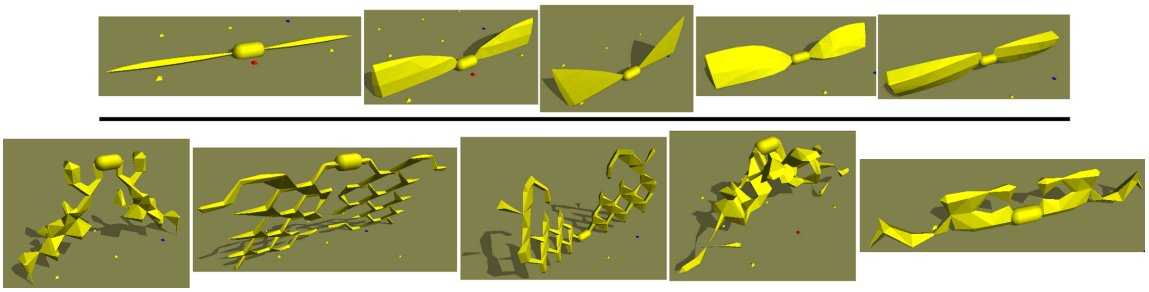


Figure 9.4: Simple and complex morphologies. The five morphologies with smallest (*top*, H_Δ values from left to right: 0.35, 0.85, 0.90, 0.91, 0.95) and largest (*bottom*, H_Δ values from left to right: 3.79, 3.81, 3.83, 3.83, 3.92) values of H_Δ across all best of trial individuals from all environments (experimental and control). The morphologies with high H_Δ values are clearly more complex than those with small H_Δ values.

Following (Page et al. 2003, Sukumar et al. 2008), the random variable x on which H is calculated is an approximation of the Gaussian curvature[8] of the points on surface. Since the bodies here are built out of triangular meshes the points at which this curvature is non-zero are precisely the vertices of the triangular mesh. Specifically, for each vertex j in a trimesh the angular deficit Φ_j is calculated as

$$\Phi_j = 2\pi - \sum_i \phi_i \tag{9.3}$$

where ϕ_i is the internal angle at j of each triangle i of which j is a vertex. This angular deficit Φ_j is directly proportional to the Gaussian curvature of that point (Page et al. 2003)[9], and so here we set $x = \Phi_j$ for calculating the entropy of curvature.

Following the calculation of Φ_j for every vertex, a PDF $p(\Phi)$ is estimated by placing the values of Φ_j into discrete bins of uniform width (Δ) and counting the number of Φ_j samples that fall into each bin. This results in a discrete set of probabilities p_i, and Eqn. 9.2 can be used to arrive at an estimate of entropy that depends on the chosen Δ, denoted here H_Δ[10].

Does H_Δ calculated in this way capture the complexity of evolved morphologies as has been demonstrated in previous work? To answer this question H_Δ is calculated for all 5000 best of trial individuals from

[8]The Gaussian curvature, K, of a point is the product of the principal curvatures, κ_1 and κ_2, of that point (Do Carmo 1976).

[9]This relationship can be derived through application of the Gauss-Bonnet theorem (Do Carmo 1976). See (Surazhsky et al. 2003) for more details.

[10]The result of this calculation is dependent on the choice of Δ. If Δ is too large the majority of samples will fall into the same bin and all information is lost. If Δ is too small then the majority of samples will fall into independent bins and H_Δ reduces to a function of the number of vertices n. In general there is no optimum Δ, and since the trimesh morphologies considered here have much fewer vertices than those of (Page et al. 2003) a correspondingly larger bin width must be used. In all calculations presented here a bin width $\Delta = \frac{\pi}{10}$ is used, chosen as a reasonable value by visually inspecting histograms of varying bin widths.

all environments (experimental and control) from CPPN-NEAT. Out of those 5000, the five morphologies which have the lowest value for this measure and the five morphologies which have the highest value for this measure are selected. Images of these morphologies are shown in Fig. 9.4. Looking at these two sets of morphologies, those with high H_Δ values appear more complex than those with low H_Δ values. In light of this observation and the previous work in this area it is concluded that H_Δ successfully captures morphological complexity.

With the knowledge that the complexity of an evolved morphology can be adequately quantified, focus shifts to how the complexity of these morphologies varies from the simple control environment to the more complex parameterized, experimental environments. From studying Fig. 9.5 it can be seen that in total the experimental environments tend to select for more complex morphologies than those which evolve in the control environment: in 18 of the experimental environments CPPN-NEAT evolves morphologies that are significantly more complex on average than those evolved in the control environment, while none of the experimental environments select for morphologies that are significantly less complex on average than those evolved in the control environment[11]. However it is unclear whether the increased complexity observed in the experimental environments is the result of an active or passive trend (McShea 1994, McShea 1996, Miconi 2008). Passive trends may result from envelope expansion without any directional bias. If there is a minimum level of complexity necessary for success, but no upper bound, than simply through random walks the existing complexity levels will increase over time. A trend will then be evident in both maximum and mean complexity. On the other hand, driven trends exhibit a consistent, directional bias. The following sections will attempt to answer this question.

9.4.2 Changes in Complexity over Evolutionary Time

In order to understand the evolutionary pressures which lead to robots that are more or less morphologically complex it is interesting to consider how morphological complexity varies over evolutionary time in different environments, and how these changes correspond to variations in fitness. For each of the environments investigated, Figs. 9.6 and 9.7 plot the mean morphological complexity and mean displacement of the current best individual in each environment taken across all trials of CPPN-NEAT for that environment. Here it can be

[11] As mentioned previously, the H_Δ values are dependent on the choice of Δ. However, this result is robust to changes in Δ. Trying all numbers of bins between 2 and 100 (so that Δ varies between 2π and $\frac{\pi}{25}$) yields a cumulative total of 1149 times that significantly more complex morphologies evolved in experimental environments compared to the control environment versus 186 times that significantly less complex morphologies evolved in experimental environments.

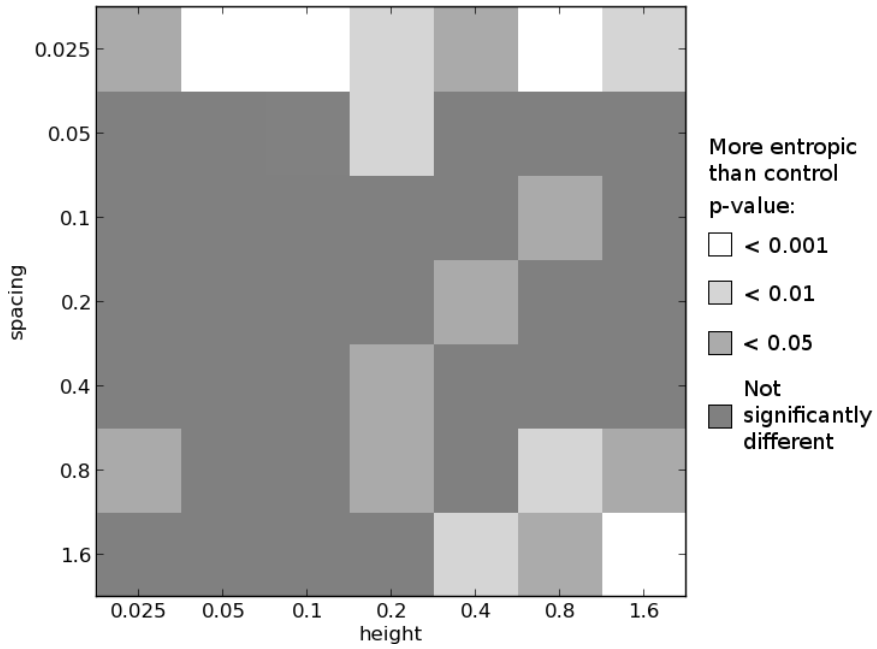


Figure 9.5: Differences in morphological complexity between experimental and control environments: single objective. Plot is created from the single objective, CPPN-NEAT results by comparing the H_{Δ} values for the best individuals from each trial in every experimental environment to the H_{Δ} values for the best individuals from each trial in the control environment. The more complex experimental environments tend to select for more complex morphologies: there are many experimental environments where significantly more complex morphologies evolve, but no experimental environments where significantly less complex morphologies evolve. All p-values calculated using the Mann-Whitney U test.

seen that while morphological complexity increases over time in all environments, in flat ground this increase stops after about generation 150, but in the vast majority of experimental, icy environments morphological complexity continues to increase along with fitness for all of evolutionary time.

These differences between flat ground and icy environments can be further demonstrated by computing correlation coefficients. For the control environment the mean fitnesses and mean morphological complexities of the best individuals (across trials) at each generation are only weakly correlated: they have a correlation coefficient (Spearman's Rho) of 0.46. Meanwhile, these quantities are much more strongly correlated in all icy environments (see Fig. 9.8). In many environments this correlation coefficient is near 1 and the mean correlation across all icy environments is 0.96. If only the second half of evolutionary time is considered these differences become even more pronounced: the correlation between mean fitness and morphological

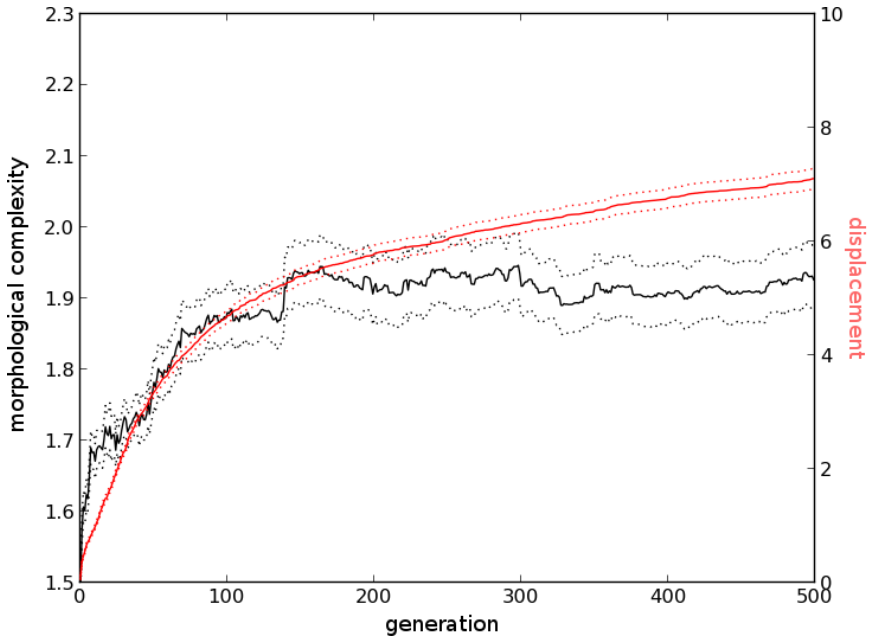


Figure 9.6: Complexity and fitness over evolutionary time in the control environment. This plot shows the mean morphological complexity (black, left-hand axis) and mean displacement (red, right-hand axis) by generation for the flat ground, control environment (solid lines). Means are taken across the 100 trials by including the most-fit individual at each generation from each trial. Dotted lines show +/- one standard error of the mean. In this environment morphological complexity plateaus at about generation 150, while fitness continues to increase.

complexity actually becomes slightly negative (-0.23) for the control environment while the mean across all experimental environments is 0.81.

9.4.3 Neutral Shadow Model

When looking only at how morphological complexity varies over evolutionary time it is unclear what change in complexity is due to selection pressure from the environment and what change is due to biases towards increasing complexity within the evolutionary algorithm itself. In order to separate the influence of these factors it is useful to compare the evolving populations to a neutral shadow model (Bedau et al. 1998, Rechtsteiner and Bedau 1999). For a generational evolutionary algorithm such as CPPN-NEAT a neutral shadow of a given experiment is equivalent to running CPPN-NEAT with the same parameters but with random selection. Fig. 9.9 shows how the morphological complexity of robots evolved in flat ground (purple), as well as a sample icy environment (blue), compare to those evolved in 100 independent trials using random selection (black) in

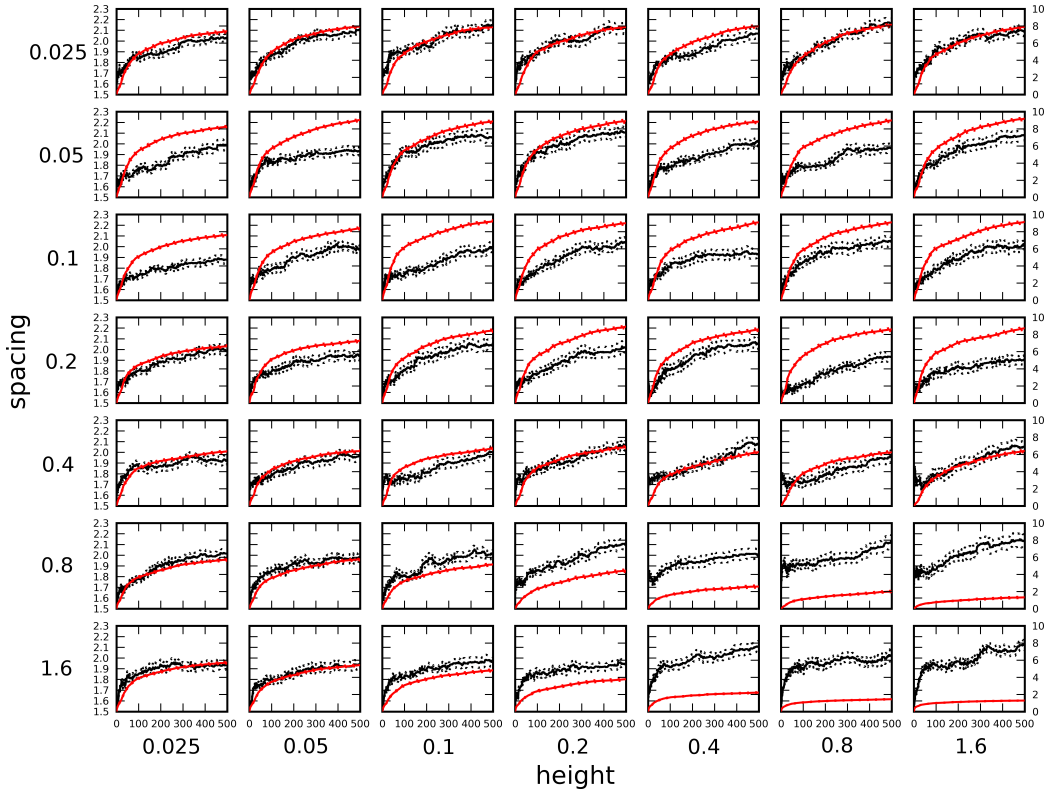


Figure 9.7: Complexity and fitness over evolutionary time in the experimental environments. Mean morphological complexity (black, left-hand axis) and mean displacement (red, right-hand axis) by generation for each of the 49 experimental environments (solid lines). Means are taken across the 100 trials of CPPN-NEAT by including the most-fit individual at each generation from each trial. Dotted lines show +/- one standard error. Unlike the control environment, complexity continues to increase along with fitness in many of the experimental environments.

which the only preference is for CPPNs that produce valid morphologies (so that there exists a morphology for which complexity can be calculated), otherwise selection is completely random. It is evident that the bias of CPPN-NEAT to produce more complex CPPN genotypes over time translates into a bias to produce more complex morphologies over time as well. In fact, random selection alone produces morphologies that are more complex than those selected for in any of the environments investigated. However, this comparison is not entirely fair. At any given generation, individuals in the random selection experiments will be the end product of many more reproduction (mutation and crossover) events than the corresponding individuals evolved for displacement, because under random selection it is unlikely that any individual will persist in

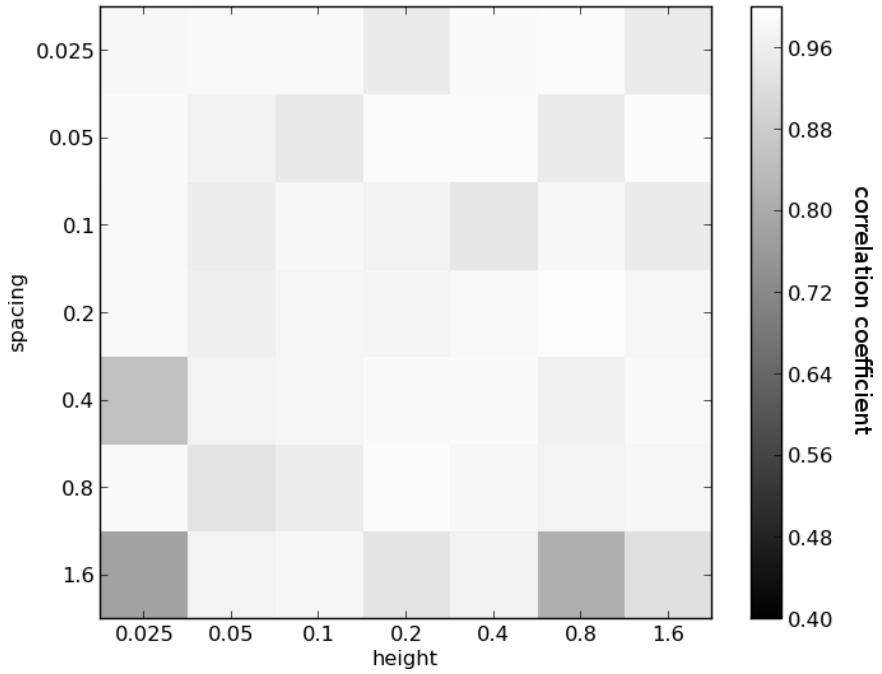


Figure 9.8: Correlation between fitness and displacement by generation. This plot shows the correlation coefficients (Spearman Rho) between mean morphological complexity and mean displacement by generation for each of the 49 experimental environments. The mean correlation coefficient across all icy environments is 0.96 vs. 0.46 in flat ground.

the population for very long. Therefore individuals in the random selection experiments will have had many more opportunities to increase the size of their genomes and hence the complexity of their morphologies.

In order to correct for this discrepancy in the number of reproduction events, alternative shadow models are employed. Specifically, neutral shadow models of both the flat ground experiments and a representative icy environment (spacing 0.025, height 0.8) are created, which control for the number of reproduction events leading to the individuals in the current population. In each of the 100 independent trial of CPPN-NEAT evolving for locomotion in both of these environments, a record of every reproduction event is created, and two alternative shadow models are created for each trial such that they maintain the same rate of reproduction. These two shadow models are detailed in Appendix B.

Both model alternatives have similar complexity curves (see Fig. 9.9) indicating that this shadow formulation is robust to whichever alternative is employed. Qualitatively they both show a much slower increase in morphological complexity (especially early on in evolution) compared to the experiments selecting for displacement, and so contrary to the naïve shadow model, both flat ground and icy environments select

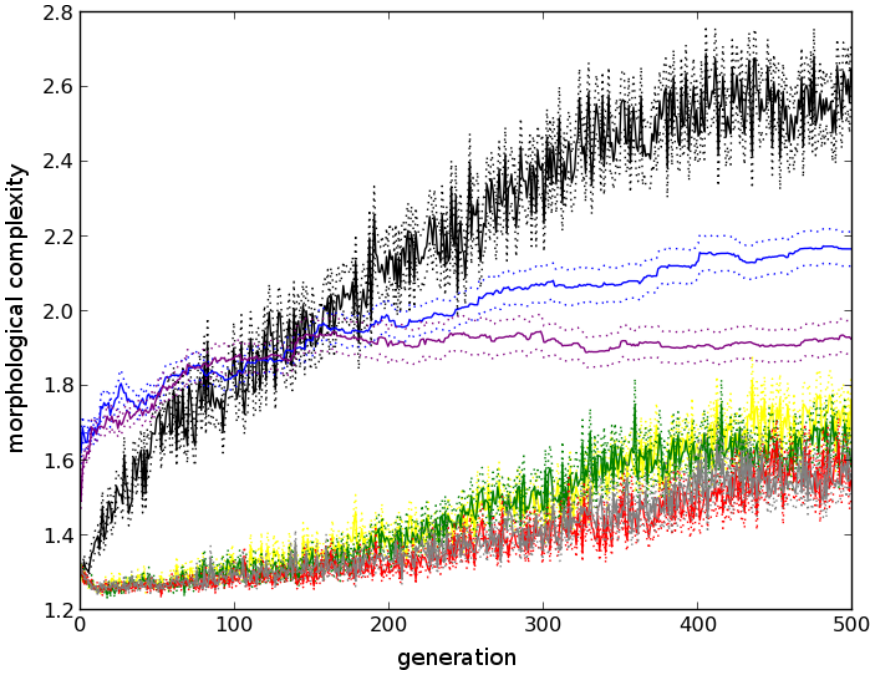


Figure 9.9: Complexity and fitness over evolutionary time vs. neutral shadows. This plot compares morphological complexity (H_{Δ}) over evolutionary time for the CPPN-NEAT experiment in the control environment (purple) and a representative, experimental, icy environment: spacing 0.025, height 0.8 (blue) along with several neutral shadow models. Solid lines denote mean (taken across all best of generation individuals from all trials in that environment) and dotted lines denote one unit of standard error. The black line depicts the naïve shadow model: completely random selection except for a preference for valid morphologies. The remaining lines depict the alternate shadow models with reproduction depths matched to the two real evolutionary experiments (see Appendix B for details). Yellow = shadow model a matched to the control environment, green = shadow model a matched to the experimental environment, red = shadow model b matched to the control environment, and gray = shadow model b matched to the experimental environment.

for increased morphological complexity beyond what would be expected in a neutral model. However, the morphological complexity plateaus in the flat ground environment at a point in evolutionary time when the shadow models are continuing to produce increasingly complex morphologies. This indicates that an appropriate level of complexity has been found for this environment and selection is actively preventing further increases in complexity. On the other hand, complexity continues to increase in this and most other icy environments (black lines in Fig. 9.7) indicating that greater morphological complexity is being actively selected for in those environments.

9.4.4 Multi-objective Selection

In an attempt to further disentangle the evolutionary pressures which lead to more complex morphologies in more complex environments, a second set of experiments is run which uses Pareto based multi-objective selection to evolve robots that can locomote in their given task environment and are as simple as possible. These multi-objective experiments seek to maximize f_1 (see Eqn. 9.1) in addition to

$$f_2 = \frac{1.0}{1.0 + H_\Delta} \tag{9.4}$$

which is strictly positive and is maximized for minimal values of H_Δ.

As was done for the single objective experiments, 100 independent trials of the NSGA-II multi-objective algorithm are run in each of the 49 icy, experimental environments as well as in the high friction, flat ground, control environment. By selecting for both maximal displacement and minimal morphological complexity these experiments should evolve robots that are no more complex than necessary to succeed in a given environment. Using this multi-objective formulation will further help elucidate the ways in which environment influences the evolution of morphological complexity.

The correlation between mean morphological complexity and mean fitness in the experimental environments shown above suggests that the increase in complexity in these environments is not a passive trend, but rather an active one. The lack of such a correlation in the control environment suggests that much of the increase in morphological complexity seen there is a result of passive processes such as the algorithm's bias toward greater complexity over time and evolutionary drift. If this is indeed the case then the differences between the complexities of morphologies evolved in the experimental environments versus the control environment are predicted to become more pronounced under multi-objective selection, which is exactly what is observed.

One problem with analyzing the results of these multi-objective experiments is that instead of producing a single best individual, each trial produces a Pareto front of non-dominated individuals representing various trade-offs between task competency and minimal complexity. An individual x is said to dominate another individual x' if x is not worse than x' on any of the objectives and x is strictly better than x' on at least one objective. The Pareto front contains all individuals y such that $\nexists x, \{x \text{ dominates } y\}$.

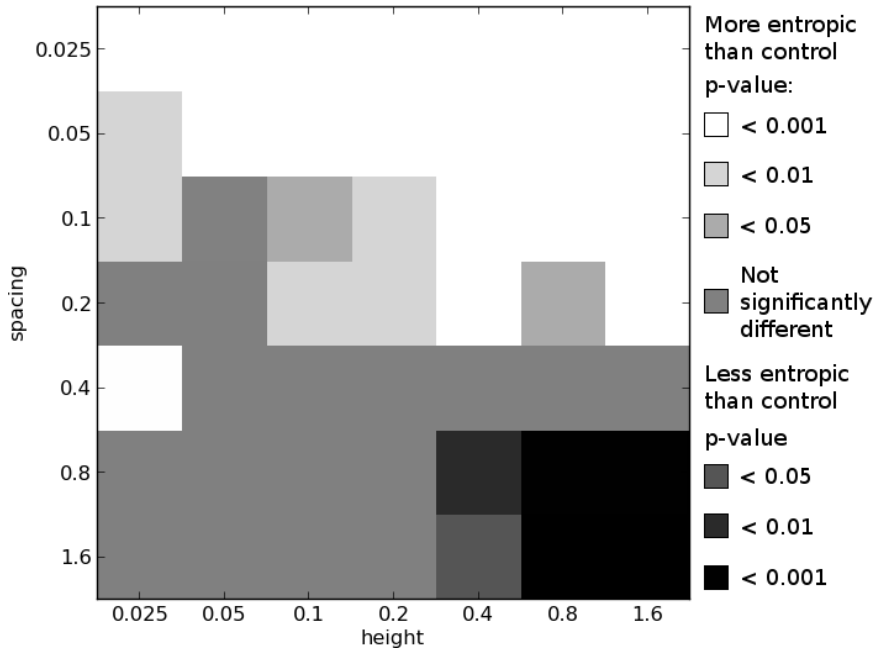


Figure 9.10: Differences in morphological complexity between experimental and control environments: multi-objective means. This plot compares the ways in which the complexity of morphologies from experimental environments differ from the complexity of morphologies evolved in the control environment under multi-objective selection. This plot is created from the multi-objective results by comparing the mean $H_\triangle$ values across each trial's final Pareto front in each experimental environment to the mean $H_\triangle$ values across each trial's final Pareto front in the control environment. See text for details. All p-values calculated using the Mann-Whitney U test.

It is not entirely clear how these Pareto fronts should be compared, as they may have different shapes (e.g. concave versus convex), and make different trade-offs between the objectives. One possibility would be to consider the best displacers in each front, but this essentially throws out the minimal complexity objective. Another possibility would be to find the knee point (Deb and Gupta 2011) on each Pareto front, but this may capture drastically different levels of competencies on different fronts, and is not well defined for convex fronts. In light of these considerations, we have adopted several methods of comparing the fronts and demonstrate that the differences in complexities between morphologies evolved in the experimental and control environment become more pronounced compared to the single objective experiments in each case (thus demonstrating that the results are not an artifact of the particular method chosen).

Fig. 9.10 shows how morphological complexity of robots evolved in each of the experimental environments under multi-objective selection differs from that of robots evolved in the control environment under

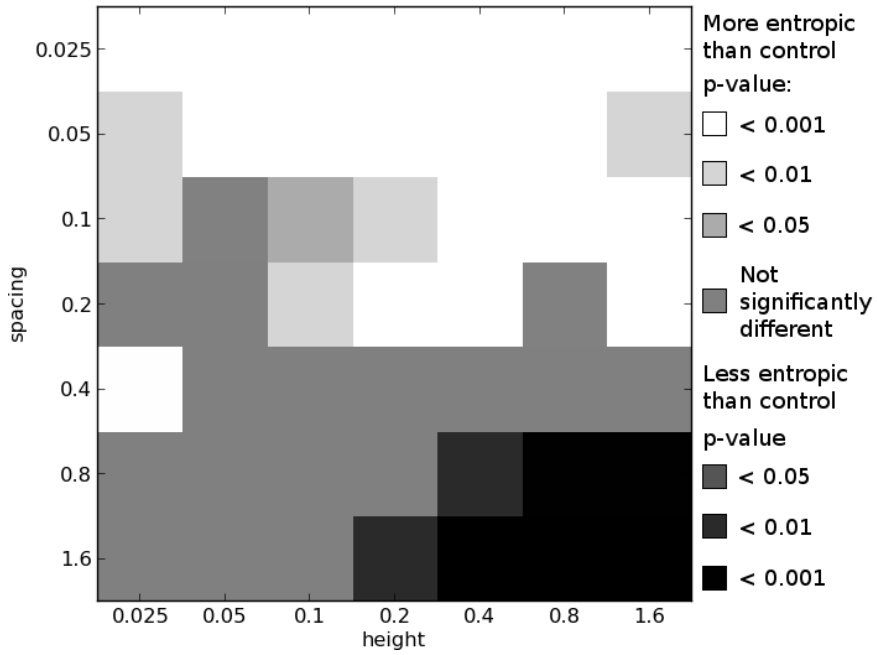


Figure 9.11: Differences in morphological complexity between experimental and control environments: multi-objective medians. This plot compares the ways in which the complexity of morphologies from experimental environments differ from the complexity of morphologies evolved in the control environment under multi-objective selection. This plot is created from the multi-objective results by comparing the H_Δ values of the median individual from each trial's final Pareto front in each experimental environment to the H_Δ value of the median individual from each trial's final Pareto front in the control environment. See text for details. All p-values calculated using the Mann-Whitney U test.

multi-objective selection. For the final Pareto front of each trial in the given environment[12] the mean morphological complexity is taken. These means (100 from each environment) are compared to the mean morphological complexity in the final Pareto front of each trial in the control environment. Fig. 9.11 presents the same comparison except it considers the median robot on each Pareto front: the robot with equal number of individuals on the front that displace less and more than it, or equivalently that are more and less complex than it. Lastly, Fig. 9.12 shows the same comparison except it considers the mean complexity of those robots in the middle half of their respective Pareto fronts. That is, the top quarter of best displacers (most complex morphologies) and the bottom quarter of worst displacers (least complex morphologies) in each front are ignored, and the means are taken across the remaining robots in each front (which should reduce the influence of any outliers).

[12]For the sake of comparing the results of the multi-objective experiments the genotypic diversity objective is ignored, and only the Pareto front consisting of the individuals not dominated on the two primary objectives is considered for each trial.

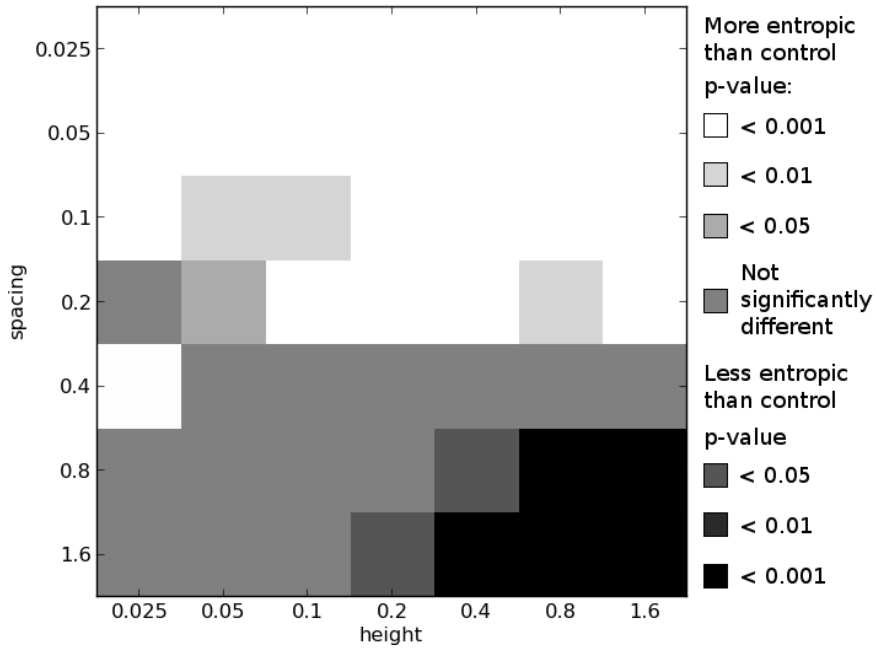


Figure 9.12: Differences in morphological complexity between experimental and control environments: multi-objective means of center halves. This plot compares the ways in which the complexity of morphologies from experimental environments differ from the complexity of morphologies evolved in the control environment under multi-objective selection. This plot is created from the multi-objective results by comparing the mean H_Δ values across the center half of each trial's final Pareto front in each experimental environment to the mean H_Δ values across the center half of each trial's final Pareto front in the control environment. See text for details. All p-values calculated using the Mann-Whitney U test.

While some differences can be observed across these plots, the general pattern is largely consistent: multi-objective selection results in more experimental environments where significantly more complex morphologies evolve in relation to the control environment than what is observed in the uni-objective case, particularly at the highest significance level (compare Figs. 9.10-9.12 with Fig. 9.5). This corroborates the hypothesis that the more complex environments actively select for increased morphological complexity, because here the inherent bias toward increased complexity is minimized by selecting for morphological simplicity.

It is also worth pointing out that in the lower right of these plots, where the environments become too difficult to succeed in (see Fig. 9.2)[13], multi-objective selection actually results in the evolution of morphologies that are significantly less complex than those that evolve to locomote on flat ground. The reason for this is that when it is not possible to evolve for greater displacement, the majority of selection is dictated by

[13]This is because in these environments the blocks are very high and the gaps are large resulting in the robots getting stuck in the gaps and not being able to get over the block in front of them.

the minimum complexity objective, and therefore simpler morphologies evolve in these environments under multi-objective selection.

9.5 Conclusions

This paper has presented a new method for evolving not only the neural system but also the physical morphology of virtual robots. Here,this method was used to investigate how different environments apply selection pressures on morphological complexity. This system differs from previous work in this domain by evolving populations of CPPNs which are used as generative encodings tp produce complex morphologies that are instantiated in virtual environments as triangular meshes. By evolving such robots in a number of different task environments and analyzing how an information theoretic measure of morphological complexity varies over evolutionary time, it was demonstrated that not only do more complex environments select for more complex morphologies, but that the more complex environments actively apply selection pressure in the direction of greater complexity above and beyond what would be expected in the absence of environmental pressure. On the other hand, it was demonstrated that not only is increased morphological complexity not necessary to succeed in a simple environment, but that simple environments can actively select against evolutionary biases towards increased complexity.

These results were corroborated with additional experiments employing multi-objective selection to not only evolve robots for task competency but also to evolve for morphological simplicity. With the imposition of this additional selection criterion the differences in morphological complexity between simple and complex environments actually becomes more pronounced. Since it is often thought that complexity comes at a cost (Fisher 1930, Orr 2000) the increased differences observed between environments in this regime may be more representative of the selection pressures present in biological systems.

While these results are illustrative of how the environment may influence the complexity of evolving morphologies, it is not likely that increased environmental complexity will select for increased morphological complexity in every case. Rather, this work has demonstrated that such a relationship does exist, and future work is needed to clarify this relationship across different environments, tasks, robots, evolutionary algorithms, and control architectures.

Additionally, a number of simplifications were made here which it may be desirable to relax in future work. By constraining the connectivity of morphological components and using a simple, open loop control architecture it was possible to largely bracket the question of neural complexity and focus on one particular aspect of morphological complexity. However, it may be desirable to investigate how many different forms of complexity evolve as a function of environment. For instance, in a recent study (Auerbach and Bongard 2012a) we demonstrated that another measure of complexity–mechanical complexity–did not increase in the same environments that selected for greater morphological complexity, but in fact mechanical complexity decreased in these environments. This result suggests that there is likely a trade-off between various forms of complexity needed to succeed in a given environment, similar to what has been shown between morphology and control (Pfeifer and Scheier 1999, Paul 2006).

To investigate these ideas further it will be interesting to allow for more complex neural architectures, more complex sensorimotor systems, and a greater diversity of materials (including 'soft' materials) to study how environments may influence the evolution of sensorial, motoric, material, mechanical, and morphological complexity of these various systems. By extending the information theoretic ideas used here for quantifying morphological complexity it is hoped that a 'common currency of bits' may be used to investigate these complexity trade-offs in a systematic manner.

9.6 References

Adamatzky, A., M. Komosinski, and S. Ulatowski (2000). Software review: Framsticks. *Kybernetes: The International Journal of Systems & Cybernetics 29*(9/10), 1344–1351.

Adami, C. (2002). What is complexity? *BioEssays 24*(12), 1085–1094.

Anderson, M. (2003). Embodied Cognition: A field guide. *Artificial Intelligence 149*(1), 91–130.

Auerbach, J. E. and J. C. Bongard (2010a). Dynamic Resolution in the Co-Evolution of Morphology and Control. In *Artificial Life XII: Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems*.

Auerbach, J. E. and J. C. Bongard (2010b). Evolving CPPNs to grow three-dimensional physical structures. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Auerbach, J. E. and J. C. Bongard (2011). Evolving Complete Robots with CPPN-NEAT: The Utility of Recurrent Connections. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Auerbach, J. E. and J. C. Bongard (2012a). On the relationship between environmental and mechanical complexity in evolved robots. In *Artificial Life XIII: Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems*.

Auerbach, J. E. and J. C. Bongard (2012b). On the relationship between environmental and morphological complexity in evolved robots. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Bedau, M. A., E. Snyder, and N. H. Packard (1998). A classification of long-term evolutionary dynamics. In C. Adami, R. K. Belew, H. Kitano, and C. Taylor (Eds.), *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*, Cambridge, MA, pp. 228–237. The MIT Press.

Beer, R. D. (2008). The dynamics of brain-body-environment systems: A status report. In P. Calvo and A. Gomila (Eds.), *Handbook of Cognitive Science: An Embodied Approach*, pp. 99–120. Elsevier.

Bongard, J. (2002a). Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, Washington, DC, USA, pp. 1872–1877. IEEE Computer Society.

Bongard, J. and R. Pfeifer (2001). Repeated structure and dissociation of genotypic and phenotypic complexity in Artificial Ontogeny. *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO)*, 829–836.

Bongard, J. C. (2002b). Evolving modular genetic regulatory networks. In *Proceedings of The IEEE 2002 Congress on Evolutionary Computation (CEC2002)*, pp. 1872–1877.

Brooks, R. (1999). *Cambrian intelligence*. MIT Press Cambridge, Mass.

Clune, J., B. Beckmann, C. Ofria, and R. Pennock (2009). Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computing*, pp. 2764–2771.

Clune, J. and H. Lipson (2011). Evolving 3D objects with a generative encoding inspired by developmental biology. In *Proceedings of the Eleventh European Conference on Artificial Life (ECAL)*, pp. 144–148.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms* (1st ed.). Wiley.

Deb, K. and S. Gupta (2011). Understanding knee points in bicriteria problems and their implications as preferred solution principles. *Engineering Optimization 43*(11), 1175–1204. Copyright of this article belongs to Taylor and Francis Group.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 6*, 182–197.

Do Carmo, M. P. (1976). *Differential Geometry of Curves and Surfaces*. Englewood Cliffs, NJ: Prentice-Hall.

Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. *Procs. of the Fourth European Conf. on Artificial Life*, 205–213.

Feldman, D. P. and J. P. Crutchfield (1998). Measures of statistical complexity: Why? *Physics Letters A 238*(45), 244 – 252.

Fisher, R. A. (1930). *The Genetical Theory of Natural Selection*. Oxford, U.K.: Oxford University Press.

Fonseca, C. and P. Fleming (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, San Francisco, CA, USA, pp. 416–423. Morgan Kaufmann Publishers Inc.

Gauci, J. and K. O. Stanley (2008). A case study on the critical role of geometric regularity in machine learning. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, AAAI'08, pp. 628–633. AAAI Press.

Gauci, J. and K. O. Stanley (2010). Autonomous evolution of topographic regularities in artificial neural networks. *Neural Comput. 22*(7), 1860–1898.

Gould, S. J. (1996, 1996). *Full House: The Spread of Excellence from Plato to Darwin*. New York, NY: Harmony Books.

Harvey, I., P. Husbands, D. Cliff, A. Thompson, and N. Jakobi (1997). Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems 20*, 205–224.

Hornby, G. S. and J. B. Pollack (2001). Body-brain co-evolution using L-systems as a generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, pp. 868–875. Morgan Kaufmann.

Jeffery, W. and D. Martasian (1998). Evolution of eye regression in the cavefish astyanax: apoptosis and the pax-6 gene. *American Zoologist 38*(4), 685–696.

Kolmogorov, A. N. (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission 1*(1), 1–7.

Komosinski, M. and A. Rotaru-Varga (2002). Comparison of different genotype encodings for simulated three-dimensional agents. *Artif. Life 7*(4), 395–418.

Lassabe, N., H. Luga, and Y. Duthen (2007). A new step for artificial creatures. In *Proceedings of 1st IEEE Conference on Artificial Life (IEEE-ALife 07)*, pp. 243–249. IEEE Press.

Lee, S. et al. (2013). Evolving gaits for physical robots with the HyperNEAT generative encoding: the benefits of simulation. *Applications of Evolutionary Computing*.

Lipson, H. and J. Pollack (2000). Automatic design and manufacture of robotic lifeforms. *Nature 406*, 974–978.

Lorensen, W. E. and H. E. Cline (1987). Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 163–169.

Lund, H. H., J. Hallam, and W. Lee (1997). Evolving robot morphology. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*, Piscataway, NJ, USA. IEEE Press.

Mautner, C. and R. Belew (2000). Evolving robot morphology and control. *Artificial Life and Robotics 4*(3), 130–136.

McCoy, J. (1977). Complexity in organic evolution. *Journal of theoretical biology 68*(3), 457.

McShea, D. W. (1991). Complexity and evolution: What everybody knows. *Biology and Philosophy 6*(3), 303–324.

McShea, D. W. (1994). Mechanisms of large-scale evolutionary trends. *Evolution 48*(6), 1747–1763.

McShea, D. W. (1996). Metazoan complexity and evolution: Is there a trend? *Evolution 50*(2), 477–492.

Miconi, T. (2008). Evolution and complexity: The double-edged sword. *Artificial Life 14*(3), 325–344.

Mouret, J. B. and S. Doncieux (2012). Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evol. Comput. 20*(1), 91–133.

Nolfi, S. and D. Floreano (2000). *Evolutionary Robotics: The Biology,Intelligence,and Technology*. Cambridge, MA, USA: MIT Press.

Orr, H. A. (2000). Adaptation and the cost of complexity. *Evolution; international journal of organic evolution 54*(1), 13–20.

Page, D., A. Koschan, S. Sukumar, B. Roui-Abidi, and M. Abidi (2003). Shape analysis algorithm based on information theory. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, Volume 1, pp. I – 229–32 vol.1.

Passy, S. (2002). Environmental randomness underlies morphological complexity of colonial diatoms. *Functional Ecology 16*(5), 690–695.

Paul, C. (2006). Morphological computation: A basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems 54*(8), 619–630.

Pfeifer, R. and J. Bongard (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press.

Pfeifer, R. and C. Scheier (1999). *Understanding intelligence*. MIT Press.

Rechtsteiner, A. and M. A. Bedau (1999). A generic neutral model for measuring excess evolutionary activity of genotypes. In *GECCO99: Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 13–17. Morgan Kaufmann.

Secretan, J., N. Beato, D. B. D'Ambrosio, A. Rodriguez, A. Campbell, J. T. Folsom-Kovarik, and K. O. Stanley (2011). Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation Journal*, 373–403.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal 27*.

Sims, K. (1994a). Evolving 3D morphology and behavior by competition. *Artif. Life 1*(4), 353–372.

Sims, K. (1994b). Evolving virtual creatures. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, New York, NY, USA, pp. 15–22. ACM.

Stanley, K., D. D'Ambrosio, and J. Gauci (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life 15*(2), 185–212.

Stanley, K. and R. Miikkulainen (2003). A taxonomy for artificial embryogeny. *Artificial Life 9*(2), 93–130.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines 8*(2), 131–162.

Stanley, K. O., B. D. Bryant, and R. Miikkulainen (2005). Real-time neuroevolution in the nero video game. *IEEE Transactions on Evolutionary Computation 9*, 653–668.

Stanley, K. O. and R. Miikkulainen (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation 10*(2), 99–127.

Sukumar, S., D. Page, A. Koschan, and M. Abidi (2008). Towards understanding what makes 3d objects appear simple or complex. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2008, Sixth IEEE Workshop on Perceptual Organization in Computer Vision (POCV)*.

Surazhsky, T., E. Magid, O. Soldea, G. Elber, and E. Rivlin (2003). A comparison of gaussian and mean curvatures estimation methods on triangular meshes. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, Volume 1, pp. 1021–1026. IEEE.

Chapter 10

Conclusion

This dissertation has presented several new techniques for evolving more complex robots, with particular focus given to methods which evolve the robots' physical forms (morphologies) in addition to their control strategies. A series of experiments was presented that demonstrated the capabilities of these techniques, analyzed their behavior, and finally employed them toward investigating the ways in which task environments may influence the evolution of complexity.

Chapter 1 presented a review of the relevant research that has been conducted to date, concentrating on studies in which aspects of morphology were placed under evolutionary control. These studies were compared along several dimensions including the scope of evolutionary control, the genotypic encodings, the phenotypic models, and the task environments investigated. While many interesting results came out of this prior work, this dissertation has culminated in a system capable of producing more complex robot morphologies (according to the morphological complexity metric employed in Chapters 7 and 9) than have previously been evolved.

Chapters 2 and 3 presented initial investigations into evolving monolithic controllers for complex robot behavior. These experiments employed scaffolding techniques to evolve controllers capable of orchestrating the dynamic motion of different body parts at different times in order to perform a complex sequence of actions. Though the robot morphologies employed in these studies were hand designed, the experiments in Chapter 3 relaxed the assumption of functional specialization. Evolution was free to determine whether

certain body parts would participate in multiple functions or specialize to only participate in a single function. This relaxation was the first step taken towards freeing morphology from the biases of human engineers.

Chapters 4 through 6 presented progressively more flexible methods for evolving structures and complete robots encoded by Compositional Pattern Producing Networks (CPPNs). These methods all combined CPPNs with a growth procedure to produce morphologies composed of spherical components. A number of benefits of this methodology were demonstrated, including the capability of CPPNs to produce the same output at multiple resolutions. This was shown to be useful for first evolving morphologies at a lower resolution, which is less costly to simulate, and then increasing the resolution over evolutionary time to eventually produce high resolution morphologies. Alternatively, as demonstrated in Chapters 5 and 6 the encoding itself can alter the growth resolution as needed so that some body components can be modeled coarsely while others can be modeled with greater detail. Moreover, it was demonstrated how allowing recurrent connections within the CPPN genomes resulted in more evolvable robots–a technique adopted in subsequent work.

Chapters 7 through 9 employed a new method for encoding robot morphologies with CPPNs that does not require an explicit growth procedure, yet is capable of modeling arbitrarily-shaped components through the use of triangular meshes. This technique was primarily employed for investigating relationships between the complexity of robot morphologies and the complexity of the environments within which they evolved. Chapter 9 in particular demonstrated how complex environments can select for complex morphologies, while simple environments select against increasing complexity. However, as demonstrated in Chapter 8, which studied the evolution of mechanical complexity, the same environments which select for increases in one form of complexity may select for decreases in another form of complexity. This demonstrates that different complexity trade-offs may be discovered by evolution.

The methods presented here represent an advancement over prior methods for evolving robot morphologies, and have been used to make several contributions to our understanding of how the environment may influence the evolution of various forms of complexity. However, there is much research still to be done if we wish to evolve robots capable of behaving safely and robustly in unstructured environments and/or to continue using simulated robots as tools for investigating evolutionary hypotheses. Some interesting and useful extensions of these methods include the ability to produce morphologies with arbitrary numbers and types of mechanical joints, morphologies with evolved material compositions (including ‘soft’ materials),

morphologies with evolved sensor distributions, and morphologies which can reconfigure themselves for different tasks.

Future work should also investigate the best ways to evolve sophisticated controllers along with morphology in order to produce robots capable of a diverse assortment of complex behaviors. It is likely that proper scaffolding techniques will be necessary to gradually ramp up behavioral competency and that neural plasticity will enable evolved robots to learn based on their experience interacting with the physical world.

Additionally, future work should investigate evolving robots in more complex environments. While the environments investigated here are more complex than the simple, high-friction, flat ground environment that has often been used for robot evolution, there are many alternate ways in which environmental complexity can be increased. Some extensions could include the presence of more and different types of obstacles, such as those with non-uniform layouts and those that may be hazardous to the robot. Another might include the presence of movable artifacts that the robots can use or manipulate. Moreover, robots could be evolved to behave in the presence of other agents: avoiding, cooperating, communicating and/or competing as necessary. These extensions will introduce new challenges, not least of which is how to effectively and efficiently simulate such scenarios in a computationally efficient manner.

One particularly promising approach to evolving robots capable of behaving in more complex environments is to use co-evolution as a form of scaffolding. By co-evolving robots and their environments it is possible to gradually increase environmental complexity without creating situations in which the robots will have no fitness gradient. Through this process it is possible to evolve robots capable of successful behavior in environments that are so complex that there is a vanishingly small probability of directly evolving successful robots. My recent (unpublished) research in this area has demonstrated this ability with a simplified model of robots and their environments. Here, 'robots' are modeled as two dimensional polygons, and 'environments' are modeled as two dimensional distributions of 'food' and 'poison' particles in various locations. By co-evolving the polygons and the placement of these particles, it is possible to evolve polygons which can enclose all the food but none of the poison in environmental configurations where direct evolution of the polygons consistently fails (see Fig. 10.1). Importantly, the polygons evolved in this research are encoded and generated by the exact same procedure used to evolve robot morphologies in Chapters 7 through 9. Because of this, it should be possible to directly transfer this approach back to evolving three dimensional morphologies and controllers in physical environments where the robots have to locomote to the food (which

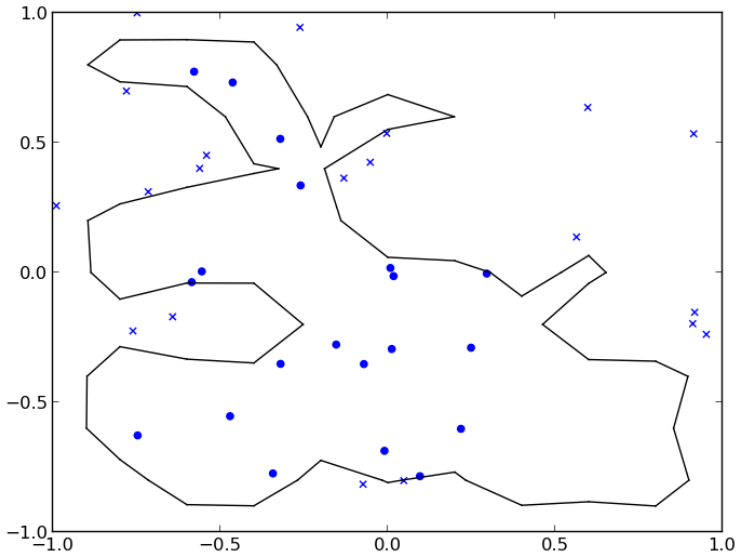


Figure 10.1: Co-evolving "robots" and environments. The robot: 2-D polygon, and the environment: placement of food (circles) and poison (Xs) particles are co-evolved such that the robot shown here successfully encapsulates all of the food particles, but none of the poison particles. When evolving directly in this environment evolution was unable to find a successful robot. This robot is produced by the same method as is used for producing 3-D morphologies in Chapters 7-9.

could also represent disaster victims) while avoiding the poison (which could also represent land mines or other hazards).

Once all of these additional capabilities have been created it will be interesting to use these techniques to further investigate the ways in which the environment can influence the evolution of various forms of complexity: sensorial, motoric, material, mechanical, and morphological. This will require developing new, information-theoretic techniques to quantify these forms of complexity. With the development of this 'common currency of bits' it should be possible to study, in a systematic manner, how these trade-offs can be struck, and thus how to evolve the proper robot for a given task's requirements and constraints.

Bibliography

Abbeel, P. and A. Ng (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1–8. ACM New York, NY, USA.

Adamatzky, A., M. Komosinski, and S. Ulatowski (2000). Software review: Framsticks. *Kybernetes: The International Journal of Systems & Cybernetics 29*(9/10), 1344–1351.

Adami, C. (2002). What is complexity? *BioEssays 24*(12), 1085–1094.

Adami, C., C. Ofria, and T. C. Collier (2000). Evolution of biological complexity. *Proceedings of the National Academy of Sciences of the United States of America 97*(9), 4463–4468.

Anderson, M. (2003). Embodied Cognition: A field guide. *Artificial Intelligence 149*(1), 91–130.

Auerbach, J. and J. C. Bongard (2009). How robot morphology and training order affect the learning of multiple behaviors. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*.

Auerbach, J. E. and J. C. Bongard (2010a). Dynamic Resolution in the Co-Evolution of Morphology and Control. In *Artificial Life XII: Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems*.

Auerbach, J. E. and J. C. Bongard (2010b). Evolving CPPNs to grow three-dimensional physical structures. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Auerbach, J. E. and J. C. Bongard (2011). Evolving Complete Robots with CPPN-NEAT: The Utility of Recurrent Connections. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Auerbach, J. E. and J. C. Bongard (2012a). On the relationship between environmental and mechanical complexity in evolved robots. In *Artificial Life XIII: Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems*.

Auerbach, J. E. and J. C. Bongard (2012b). On the relationship between environmental and morphological complexity in evolved robots. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

Bailly-Salins, I. and H. Luga (2007). Artistic 3d object creation using artificial life paradigms. In *SG '07: Proceedings of the 8th international symposium on Smart Graphics*, Berlin, Heidelberg, pp. 135–145. Springer-Verlag.

Balakrishnan, K. and V. Honavar (1996). On sensor evolution in robotics. In *GECCO '96: Proceedings of the First Annual Conference on Genetic Programming*, Cambridge, MA, USA, pp. 455–460. MIT Press.

Ball, P. (2009). *Branches: Nature's Patterns: A Tapestry in Three Parts*. Oxford University Press.

Bedau, M. A. (1998). Four puzzles about life. *Artificial Life 4*(2), 125 – 140.

Bedau, M. A., E. Snyder, and N. H. Packard (1998). A classification of long-term evolutionary dynamics. In C. Adami, R. K. Belew, H. Kitano, and C. Taylor (Eds.), *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*, Cambridge, MA, pp. 228–237. The MIT Press.

Beer, R. (2003). The dynamics of active categorical perception in an evolved model agent. *Adaptive Behavior 11*(4), 209.

Beer, R. D. (1990). *Intelligence as adaptive behavior: an experiment in computational neuroethology*. San Diego, CA, USA: Academic Press Professional, Inc.

Beer, R. D. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. In P. Maes, M. Mataric, J. Meyer, J. Pollack, and S. Wilson (Eds.), *From Animals to Animats 4: Proceedings*

of the Fourth International Conference on the Simulation of Adaptive Behaviour, pp. 421–429. MIT Press.

Beer, R. D. (2006). Parameter space structure of continuous-time recurrent neural networks. *Neural Comp. 18*(12), 3009–3051.

Beer, R. D. (2008). The dynamics of brain-body-environment systems: A status report. In P. Calvo and A. Gomila (Eds.), *Handbook of Cognitive Science: An Embodied Approach*, pp. 99–120. Elsevier.

Bongard, J. (2002a). Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, Washington, DC, USA, pp. 1872–1877. IEEE Computer Society.

Bongard, J. (2008a). Behavior chaining: incremental behavioral integration for evolutionary robotics. In S. Bullock, J. Noble, R. Watson, and M. A. Bedau (Eds.), *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pp. 64–71. MIT Press, Cambridge, MA.

Bongard, J. and H. Lipson (2007). Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Science 104*(24), 9943–9948.

Bongard, J. and R. Pfeifer (2001). Repeated structure and dissociation of genotypic and phenotypic complexity in Artificial Ontogeny. *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO)*, 829–836.

Bongard, J. C. (2002b). Evolving modular genetic regulatory networks. In *Proceedings of The IEEE 2002 Congress on Evolutionary Computation (CEC2002)*, pp. 1872–1877.

Bongard, J. C. (2008b). Behavior chaining: incremental behavior integration for evolutionary robotics. In M. Press (Ed.), *Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pp. 64–71.

Bongard, J. C. and C. Paul (2000). Investigating morphological symmetry and locomotive efficiency using virtual embodied evolution. In *From Animals to Animats: The Sixth International Conference on the Simulation of Adaptive Behaviour*, pp. 420–429. MIT Press.

Bongard, J. C. and C. Paul (2001). Making evolution an offer it can't refuse: Morphology and the extradimensional bypass. In *ECAL '01: Proceedings of the 6th European Conference on Advances in Artificial Life*, London, UK, pp. 401–412. Springer-Verlag.

Bongard, J. C. and R. Pfeifer (2003). Evolving complete agents using artificial ontogeny. In *Morphofunctional Machines: The New Species (Designing Embodied Intelligence*, pp. 237–258. Springer-Verlag.

Bonner, J. T. (1988). *The Evolution of Complexity by Means of Natural Selection*. Princeton, N.J.: Princeton University Press.

Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. MIT Press.

Brooks, R. (1986). A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of [legacy, pre - 1988] 2*(1), 14–23.

Brooks, R. (1999). *Cambrian intelligence*. MIT Press Cambridge, Mass.

Buason, G., N. Bergfeldt, and T. Ziemke (2005). Brains, bodies, and beyond: Competitive co-evolution of robot controllers, morphologies and environments. *Genetic Programming and Evolvable Machines 6*(1), 25–51.

Bugajska, M. D. and A. C. Schultz (2000). Co-evolution of form and function in the design of autonomous agents: Micro air vehicle project. In *Proc. Workshop on Evolution of Sensors GECCO 2000, IEEE*, pp. 240–244.

Calabretta, R., S. Nolfi, D. Parisi, and G. P. Wagner (1998). Emergence of functional modularity in robots. In *Proceedings of the fifth international conference on simulation of adaptive behavior on From animals to animats 5*, Cambridge, MA, USA, pp. 497–504. MIT Press.

Calabretta, R., S. Nolfi, D. Parisi, and G. P. Wagner (2000). Duplication of modules facilitates the evolution of functional specialization. *Artificial Life 6*(1), 69–84.

Carriker, W., P. Khosla, and B. Krogh (1991). Path planning for mobile manipulators for multiple task execution. *IEEE Transactions on Robotics and Automation*, 403 – 408.

CGAFaq (2010). Evenly distributed points on sphere – cgafaq. [Online; http://cgafaq.info/wiki/Evenly_distributed_points_on_sphere accessed 25-January-2010].

Chaumont, N., R. Egli, and C. Adami (2007). Evolving virtual creatures and catapults. *Artif. Life 13*(2), 139–157.

Chella, A., H. Dindo, F. Matraxia, and R. Pirrone (2007). Real-time visual grasp synthesis using genetic algorithms and neural networks. In R. Basili and M. T. Pazienza (Eds.), *AI*IA*, Volume 4733 of *Lecture Notes in Computer Science*, pp. 567–578. Springer.

Clark, A., J. Moore, J. Wang, X. Tan, and P. McKinley (2012). Evolutionary design and experimental validation of a flexible caudal fin for robotic fish. In *Proceedings of the Thirteenth International Conference on the Synthesis and Simulation of Living Systems*, East Lansing, Michigan, USA. Best Paper Award.

Clune, J., B. Beckmann, C. Ofria, and R. Pennock (2009). Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computing*, pp. 2764–2771.

Clune, J. and H. Lipson (2011). Evolving 3D objects with a generative encoding inspired by developmental biology. In *Proceedings of the Eleventh European Conference on Artificial Life (ECAL)*, pp. 144–148.

Clune, J., R. T. Pennock, and C. Ofria (2009). The sensitivity of HyperNEAT to different geometric representations of a problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

Collins, S., A. Ruina, R. Tedrake, and M. Wisse (2005). Efficient bipedal robots based on passive-dynamic walkers. *Science 307*(5712), 1082–1085.

Cussat-Blanc, S. and J. Pollack (2012). A cell-based developmental model to generate robot morphologies. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, New York, NY, USA, pp. 537–544. ACM.

Dawkins, R. (1986). *The Blind Watchmaker: Why the evidence of evolution reveals a universe without design*. WW Norton & Company.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms* (1st ed.). Wiley.

Deb, K. and S. Gupta (2011). Understanding knee points in bicriteria problems and their implications as preferred solution principles. *Engineering Optimization 43*(11), 1175–1204. Copyright of this article belongs to Taylor and Francis Group.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 6*, 182–197.

Dellaert, F. and R. Beer (1994). Toward an evolvable model of development for autonomous agent synthesis. *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*.

Dellaert, F., R. D. Beer, and A. D. Beer (1996). A developmental model for the evolution of complete autonomous agents. In *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pp. 393–401. MIT Press.

Do Carmo, M. P. (1976). *Differential Geometry of Curves and Surfaces*. Englewood Cliffs, NJ: Prentice-Hall.

Dollar, A. and R. Howe (2007). Simple, robust autonomous grasping in unstructured environments. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pp. 4693–4700.

Dorigo, M. and M. Colombetti (1994). Robot shaping: Developing situated agents through learning. *Artificial Intelligence 70*(2), 321–370.

Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. *Procs. of the Fourth European Conf. on Artificial Life*, 205–213.

Eggenberger, P. (2003). Evolving morphologies and neural controllers based on the same underlying principle: Specific ligand-receptor interactions. In F. Hara and R. Pfeifer (Eds.), *Morpho-functional machines: the new species; designing embodied intelligence*, pp. 217–236. Tokyo, Japan: Springer.

Endo, K., T. Maeno, and H. Kitano (2002). Co-evolution of morphology and walking pattern of biped humanoid robot using evolutionary computation. consideration of characteristic of the servomotors. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, Volume 3, pp. 2678 – 2683 vol.3.

Feldman, D. P. and J. P. Crutchfield (1998). Measures of statistical complexity: Why? *Physics Letters A 238*(45), 244 – 252.

Fernandez Jr., J. J. and I. D. Walker (1999). A biologically inspired fitness function for robotic grasping. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith (Eds.), *Proc. of the Genetic and Evolutionary Computation Conf. GECCO-99*, San Francisco, CA, pp. 1517–1522. Morgan Kaufmann.

Fisher, R. A. (1930). *The Genetical Theory of Natural Selection*. Oxford, U.K.: Oxford University Press.

Fonseca, C. and P. Fleming (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, San Francisco, CA, USA, pp. 416–423. Morgan Kaufmann Publishers Inc.

Foss, J., F. Moss, and J. Milton (1997). Noise, multistability, and delayed recurrent loops. *Physical Review E 55*(4), 4536+.

Fuller, R. B. (1961). Tensegrity. *Portfolio Artnews Annual 4*, 112–127.

Funes, P. and J. Pollack (1997). Computer evolution of buildable objects. *Fourth European Conference on Artificial Life*, 358–367.

Gauci, J. and K. O. Stanley (2008). A case study on the critical role of geometric regularity in machine learning. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, AAAI'08, pp. 628–633. AAAI Press.

Gauci, J. and K. O. Stanley (2010). Autonomous evolution of topographic regularities in artificial neural networks. *Neural Comput. 22*(7), 1860–1898.

Gould, S. J. (1996, 1996). *Full House: The Spread of Excellence from Plato to Darwin*. New York, NY: Harmony Books.

Grefenstette, J. J. (1984). The user's guide to GENESIS. *Technical Report CS-83-11, Computer Science Department, Vanderbilt University*.

Grefenstette, J. J. (1991). The user's guide to SAMUEL, version 1.3. *NRL Memorandum Report 6820, Naval Research Laboratory*.

Hansen, N. and A. Ostermeier (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation 9*(2), 159–195.

Hara, F. and K. Kikuchi (2003). Design principles of morpho-functional machines. In F. Hara and R. Pfeifer (Eds.), *Morpho-functional machines: the new species; designing embodied intelligence*, pp. 259–286. Tokyo, Japan: Springer.

Harvey, I., P. Husbands, D. Cliff, A. Thompson, and N. Jakobi (1997). Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems 20*, 205–224.

Heinen, M. R. and F. S. Osório (2009). Evolving morphologies and gaits of physically realistic simulated robots. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, New York, NY, USA, pp. 1161–1165. ACM.

Hiller, J. and H. Lipson (2012). Automatic design and manufacture of soft robots. *Robotics, IEEE Transactions on 28*(2), 457–466.

Hiller, J. D. and H. Lipson (2010). Evolving amorphous robots. In *Artificial Life XII: Proceedings of the Twelfth International Conference on the Simulation and Synthesis of Living Systems*, Cambridge, MA, pp. 717–724. MIT Press.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press.

Hornby, G., S. Takamura, T. Yamamoto, and M. Fujita (2005). Autonomous evolution of dynamic gaits with two quadruped robots. *Robotics, IEEE Transactions on 21*(3), 402–410.

Hornby, G. S., H. Lipson, and J. B. Pollack (2003). Generative representations for the automated design of modular physical robots. *IEEE Transactions on Robotics and Automation 19*, 703–719.

Hornby, G. S. and J. B. Pollack (2001a). The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001)*, pp. 600–607. IEEE Press.

Hornby, G. S. and J. B. Pollack (2001b). Body-brain co-evolution using L-systems as a generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, San Francisco, California, USA, pp. 868–875. Morgan Kaufmann.

Hornby, G. S. and J. B. Pollack (2001c). Evolving L-systems to generate virtual creatures. *Computers and Graphics 25*(6), 1041–1048.

Inamura, T., I. Toshima, and H. Tanie (2004). Embodied symbol emergence based on mimesis theory. *International Journal of Robotics Research 23*(4), 363–377.

Izquierdo, E. and T. Buhrmann (2008). Analysis of a dynamical recurrent neural network evolved for two qualitatively different tasks: walking and chemotaxis. In S. Bullock, J. Noble, R. Watson, and M. A. Bedau (Eds.), *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pp. 257–264. MIT Press, Cambridge, MA.

Jakobi, N. et al. (1998). The minimal simulation approach to evolutionary robotics. *Proceedings of ER 98*.

Jeffery, W. and D. Martasian (1998). Evolution of eye regression in the cavefish astyanax: apoptosis and the pax-6 gene. *American Zoologist 38*(4), 685–696.

Joachimczak, M. and B. Wróbel (2012). Co-evolution of morphology and control of soft-bodied multicellular animats. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, New York, NY, USA, pp. 561–568. ACM.

Kashtan, N. and U. Alon (2005). Spontaneous evolution of modularity and network motifs. *Proc Natl Acad Sci U S A*.

Klein, J. (2003). breve: a 3d environment for the simulation of decentralized systems and artificial life. In *ICAL 2003: Proceedings of the eighth international conference on Artificial life*, Cambridge, MA, USA, pp. 329–334. MIT Press.

Kolmogorov, A. N. (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission 1*(1), 1–7.

Komosinski, M. (2000). The world of framsticks: Simulation, evolution, interaction. In *VW '00: Proceedings of the Second International Conference on Virtual Worlds*, London, UK, pp. 214–224. Springer-Verlag.

Komosinski, M. and A. Rotaru-Varga (2002). Comparison of different genotype encodings for simulated three-dimensional agents. *Artif. Life 7*(4), 395–418.

Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Boston, MA: MIT Press.

Krčah, P. (2008). Towards efficient evolution of morphology and control. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, New York, NY, USA, pp. 287–288. ACM.

Lamarck, J. (1809). *Philosophie zoologique: ou Exposition des considérations relative à l'histoire naturelle des animaux*. Philosophic zoologique: ou Exposition des considérations relative à l'histoire naturelle des animaux. Dentu et L'Auteur.

Lassabe, N., H. Luga, and Y. Duthen (2006). Evolving creatures in virtual ecosystems. In Z. Pan, A. D. Cheok, M. Haller, R. W. H. Lau, H. Saito, and R. Liang (Eds.), *ICAT*, Volume 4282 of *Lecture Notes in Computer Science*, pp. 11–20. Springer.

Lassabe, N., H. Luga, and Y. Duthen (2007). A new step for artificial creatures. In *Proceedings of 1st IEEE Conference on Artificial Life (IEEE-ALife 07)*, pp. 243–249. IEEE Press.

Lee, S. et al. (2013). Evolving gaits for physical robots with the HyperNEAT generative encoding: the benefits of simulation. *Applications of Evolutionary Computing*.

Lee, W., J. Hallam, H. H. Lund, and E. U. K (1996). A hybrid GP/GA approach for co-evolving controllers and robot bodies to achieve fitness-specified tasks. In *Proceedings of IEEE 3rd International Conference on Evolutionary Computation*, pp. 384–389. IEEE Press.

Lehman, J. and K. O. Stanley (2011a). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation 19*(2), 189–223.

Lehman, J. and K. O. Stanley (2011b). Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2011)*. ACM.

Lenski, R. E., C. Ofria, R. T. Pennock, and C. Adami (2003). The evolutionary origin of complex features. *Nature 423*, 139–144.

Lichtensteiger, L. and P. Eggenberger (1999). Evolving the morphology of a compound eye on a robot. In *Proceedings of the Third European Workshop on Advanced Mobile Robots (Eurobot 99*, pp. 127–134. IEEE Press.

Lindenmayer, A. (1968). Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology 18*(3), 300–315.

Lipson, H. and J. Pollack (2000). Automatic design and manufacture of robotic lifeforms. *Nature 406*, 974–978.

Lipson, H., J. Pollack, and N. Suh (2002). On The Origin of Modular Variation. *Evolution 56*(8), 1549–1556.

Lorensen, W. E. and H. E. Cline (1987). Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 163–169.

Luga, H. (2008). Automatic generation of behaviors, morphologies and shapes of virtual entities. In D. Plemenos and G. Miaoulis (Eds.), *Artificial Intelligence Techniques for Computer Graphics*, Volume 159 of *Studies in Computational Intelligence*, pp. 103–121. Springer.

Lund, H. H. (2003). Co-evolving control and morphology with lego robots. In F. Hara and R. Pfeifer (Eds.), *Morpho-functional machines: the new species; designing embodied intelligence*, pp. 59–79. Tokyo, Japan: Springer.

Lund, H. H., J. Hallam, and W. Lee (1997). Evolving robot morphology. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*, Piscataway, NJ, USA. IEEE Press.

Lund, H. H. and O. Miglino (1998). Evolving and breeding robots. In *Proceedings of the First European Workshop on Evolutionary Robotics*, London, UK, pp. 192–210. Springer-Verlag.

Macinnes, I. (2003). Visually guided physically simulated agents with evolved morphologies. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler (Eds.), *ECAL*, Volume 2801 of *Lecture Notes in Computer Science*, pp. 821–828. Springer.

Macinnes, I. and E. D. Paolo (2004). Crawling out of the simulation: Evolving real robot morphologies using cheap reusable modules. In *Artificial Life IX: Proc. Ninth Intl. Conf. on the Simulation and Synthesis of Life*, pp. 94–99. MIT Press.

Marbach, D. and A. Ijspeert (2004). Co-evolution of Configuration and Control for Homogenous Modular Robots. In F. Groen (Ed.), *Proceedings of the Eighth Conference on Intelligent Autonomous Systems (IAS8)*, pp. 712–719. IOS Press.

Mark, A., D. Polani, and T. Uthmann (1998). A framework for sensor evolution in a population of braitenberg vehicle-like agents (poster). In *ALIFE: Proceedings of the sixth international conference on Artificial life*, Cambridge, MA, USA, pp. 428–432. MIT Press.

Mautner, C. and R. Belew (2000). Evolving robot morphology and control. *Artificial Life and Robotics 4*(3), 130–136.

Maynard Smith, J. (1988). Evolutionary progress and levels of selection. *Evolutionary Progress*.

McCoy, J. (1977). Complexity in organic evolution. *Journal of theoretical biology 68*(3), 457.

McShea, D. (2005). The evolution of complexity without natural selection, a possible large-scale trend of the fourth kind. *Paleobiology 31*(sp5), 146–156.

McShea, D. W. (1991). Complexity and evolution: What everybody knows. *Biology and Philosophy 6*(3), 303–324.

McShea, D. W. (1994). Mechanisms of large-scale evolutionary trends. *Evolution 48*(6), 1747–1763.

McShea, D. W. (1996). Metazoan complexity and evolution: Is there a trend? *Evolution 50*(2), 477–492.

McShea, D. W. and R. N. Brandon (2010). *Biology's First Law: The Tendency for Diversity and Complexity to Increase in Evolutionary Systems*. Chicago, IL: University of Chicago Press.

Miconi, T. (2008a). Evolution and complexity: The double-edged sword. *Artificial Life 14*(3), 325–344.

Miconi, T. (2008b). In silicon no one can hear you scream: Evolving fighting creatures. In M. O'Neill, L. Vanneschi, S. Gustafson, A. Esparcia-Alcázar, I. D. Falco, A. D. Cioppa, and E. Tarantino (Eds.), *EuroGP*, Volume 4971 of *Lecture Notes in Computer Science*, pp. 25–36. Springer.

Miconi, T. and A. Channon (2005). A virtual creatures model for studies in artificial evolution. In *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*.

Miglino, O., O. Gigliotta, A. Rega, and M. Ponticorvo (2007). Introducing human trainers in evolutionary robotics. In *Proceedings of WIVACE 2007*, pp. 1–7.

Mondada, F., E. Franzi, and P. Ienne (1994). Mobile robot miniaturisation: A tool for investigation in control algorithms. *Experimental Robotics III*, 501–513.

Moore, J. M. and P. K. McKinley (2012). Evolving flexible joint morphologies. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, New York, NY, USA, pp. 145–152. ACM.

Moravec, H., J. Easudes, and F. Dellaert (1996). Fractal branching ultra-dexterous robots (bush robots). Technical report, NASA Advanced Concepts Research Project. PR-Number 10-86888.

Mouret, J. B. and S. Doncieux (2012). Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evol. Comput. 20*(1), 91–133.

Nelson, A. L., G. J. Barlow, and L. Doitsidis (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems 57*(4), 345–370.

Noë, A. (2004). *Action In Perception*. MIT Press.

Nolfi, S. and D. Floreano (2000). *Evolutionary Robotics: The Biology,Intelligence,and Technology*. Cambridge, MA, USA: MIT Press.

Okada, M. and Y. Nakamura (2004). Design of the continuous symbol space for the intelligent robots using the dynamics-based information processing. *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on 4*, 3201–3206 Vol.4.

O'Kelly, M. J. and K. Hsiao (2004). Evolving simulated mutually perceptive creatures for combat. In *Artificial Life IX: Proc. Ninth Intl. Conf. on the Simulation and Synthesis of Life*, pp. 113–118. MIT Press.

Orr, H. A. (2000). Adaptation and the cost of complexity. *Evolution; international journal of organic evolution 54*(1), 13–20.

Page, D., A. Koschan, S. Sukumar, B. Roui-Abidi, and M. Abidi (2003). Shape analysis algorithm based on information theory. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, Volume 1, pp. I – 229–32 vol.1.

Parker, G. B., A. S. Anev, and D. Duzevik (2003). Evolving towers in a 3-dimensional simulated environment. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, pp. 1137–1144. IEEE Press.

Parker, G. B., D. Duzevik, A. S. Anev, and R. Georgescu (2007). Morphological evolution of dynamic structures in a 3-dimensional simulated environment. In *CIRA*, pp. 534–540.

Parker, G. B. and P. J. Nathan (2007). Co-evolution of sensor morphology and control on a simulated legged robot. In *CIRA*, pp. 516–521.

Passy, S. (2002). Environmental randomness underlies morphological complexity of colonial diatoms. *Functional Ecology 16*(5), 690–695.

Paul, C. (2006). Morphological computation: A basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems 54*(8), 619–630.

Paul, C., H. Lipson, and F. J. V. Cuevas (2005). Evolutionary form-finding of tensegrity structures. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, New York, NY, USA, pp. 3–10. ACM.

Pfeifer, R. and J. Bongard (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press.

Pfeifer, R. and C. Scheier (1999). *Understanding intelligence*. MIT Press.

Pollack, J. B., H. Lipson, G. Hornby, and P. Funes (2001). Three generations of automatically designed robots. *Artif. Life 7*(3), 215–223.

Pugh, A. (1976). *An introduction to tensegrity*. Univ of California Press.

Rechenberg, I. (1973). Evolutionsstrategie–optimierung technisher systeme nach prinzipien der biologischen evolution.

Rechtsteiner, A. and M. A. Bedau (1999). A generic neutral model for measuring excess evolutionary activity of genotypes. In *GECCO99: Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 13–17. Morgan Kaufmann.

Reil, T. and P. Husbands (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. *Evolutionary Computation, IEEE Transactions on 6*(2), 159–168.

Rieffel, J., F. Valero-Cuevas, and H. Lipson (2009). Automated discovery and optimization of large irregular tensegrity structures. *Computers & Structures 87*(5-6), 368 – 379.

Rieffel, J. A., F. J. Valero-Cuevas, and H. Lipson (2010). Morphological communication: exploiting coupled dynamics in a complex mechanical structure to achieve locomotion. *J R Soc Interface 7*(45), 613–21.

Römmerman, M., D. Kühn, and F. Kirchner (2009). Robot design for space missions using evolutionary computation. In *CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, Piscataway, NJ, USA, pp. 2098–2105. IEEE Press.

Russell, S. J. and P. Norvig (2002). *Artificial Intelligence: A Modern Approach* (Second ed.). Prentice Hall.

Saff, E. and A. Kuijlaars (1997). Distributing many points on a sphere. *The Mathematical Intelligencer 19*(1), 5–11.

Saksida, L., S. Raymond, and D. S. Touretzky (1997). Shaping robot behavior using principles from instrumental conditioning. *Robotics and Autonomous Systems 22*, 231–249.

Schramm, L., Y. Jin, and B. Sendhoff (2011). Emerged coupling of motor control and morphological development in evolution of multi-cellular animats. In *Proceedings of the 10th European conference on Advances in artificial life: Darwin meets von Neumann - Volume Part I*, ECAL'09, Berlin, Heidelberg, pp. 27–34. Springer-Verlag.

Secretan, J., N. Beato, D. B. D'Ambrosio, A. Rodriguez, A. Campbell, J. T. Folsom-Kovarik, and K. O. Stanley (2011). Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation Journal*, 373–403.

Seraji, H. (1998). A unified approach to motion control of mobile manipulators. *The International Journal of Robotics Research 17*(2), 107–118.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal 27*.

Shim, Y.-S. and C.-H. Kim (2003). Generating flying creatures using body-brain co-evolution. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Aire-la-Ville, Switzerland, Switzerland, pp. 276–285. Eurographics Association.

Silverman, B. W. (1986). *Density estimation: for statistics and data analysis*. London.

Sims, K. (1994a). Evolving 3D morphology and behavior by competition. *Artif. Life 1*(4), 353–372.

Sims, K. (1994b). Evolving virtual creatures. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, New York, NY, USA, pp. 15–22. ACM.

Singh, S. P. (1992). Transfer of learning across sequential tasks. *Machine Learning 8*, 323–339.

Spector, L., J. Klein, and M. Feinstein (2007). Division blocks and the open-ended evolution of development, form, and behavior. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, New York, NY, USA, pp. 316–323. ACM.

Stanley, K., D. D'Ambrosio, and J. Gauci (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life 15*(2), 185–212.

Stanley, K. and R. Miikkulainen (2003). A taxonomy for artificial embryogeny. *Artificial Life 9*(2), 93–130.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines 8*(2), 131–162.

Stanley, K. O., B. D. Bryant, and R. Miikkulainen (2005). Real-time neuroevolution in the nero video game. *IEEE Transactions on Evolutionary Computation 9*, 653–668.

Stanley, K. O. and R. Miikkulainen (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation 10*(2), 99–127.

Suh, N. P. (1990). *The Principles of Design (Oxford Series on Advanced Manufacturing)*. Oxford University Press.

Sukumar, S., D. Page, A. Koschan, and M. Abidi (2008). Towards understanding what makes 3d objects appear simple or complex. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2008, Sixth IEEE Workshop on Perceptual Organization in Computer Vision (POCV)*.

Surazhsky, T., E. Magid, O. Soldea, G. Elber, and E. Rivlin (2003). A comparison of gaussian and mean curvatures estimation methods on triangular meshes. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, Volume 1, pp. 1021–1026. IEEE.

Sutton, R. and A. Barto (1999). Reinforcement learning. *Journal of Cognitive Neuroscience 11*(1), 126–134.

Tassa, Y., T. Erez, and W. Smart (2008). Receding horizon differential dynamic programming. *Advances in Neural Information Processing Systems 20*, 1465–1472.

Taylor, T. and C. Massey (2000). Recent developments in the evolution of morphologies and controllers for physically simulated creatures. *Artif. Life 7*(1), 77–87.

Tedrake, R., T. Zhang, and H. Seung (2005). Learning to walk in 20 minutes. In *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, Yale University, New Haven, CT.

Ventrella, J. (1994). Explorations in the emergence of morphology and locomotion behavior in animated characters. In *Artificial Life IV proceedings*. MIT Press.

Verbancsics, P. and K. O. Stanley (2011). Constraining connectivity to encourage modularity in HyperNEAT. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*.

von Haller, B., A. Ijspeert, and D. Floreano (2005). Co-evolution of Structures and Controllers for Neubot Underwater Modular Robots. In M. S. Capcarrere, A. A. Freitas, P. J. Bentley, C. G. Johnson, J. Timmis (eds) (Ed.), *Advances in Artificial Life*, Lecture Notes in Artificial Intelligence, Berlin. Springer Verlag.

Wagner, G. and L. Altenberg (1996). Perspective: Complex adaptations and the evolution of evolvability. *Evolution 50*(3), 967–976.

Wampler, K. and Z. Popović (2009). Optimal gait and form for animal locomotion. *ACM Trans. Graph. 28*(3), 1–8.

Watson, R. A., S. G. Ficici, and J. B. Pollack (1999). Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *Congress on Evolutionary Computation*, pp. 335–342. IEEE Press.

Wood, D., J. Bruner, and G. Ross (1976). The role of tutoring in problem solving. *J Child Psychol Psychiatry 17*(2), 89–100.

Appendix A: Parameters

Table A.1: Evolutionary Algorithm Parameters.

Parameter Name	**Value**
Population Size	150
Max Generations	500
Mutate Add Node Probability	0.03
Mutate Add Link Probability	0.05
Mutate Demolish Link Probability	0.00
Mutate Link Weights Probability	0.8
Mutate Only Probability	0.25
Mutate Link Probability	0.1
Allow Add Node To Recurrent Connection	No
Mutation Power	2.5
Adult Link Age	18.0
Allow Recurrent Connections	Yes
Allow Self Recurrent Connections	No
Force Copy Generation Champion	Yes
Link Gene Minimum Weight For Phenotype	0.0

Table A.2: Encoding Parameters.

Parameter Name	**Value**
Number of CPPN Update Iterations	10
CPPN Activation functions	Signed cosine, Gaussian, and sigmoid
Matter Threshold	0.5
Amplitude Range	$\left[\frac{\pi}{4}, \frac{3\pi}{4}\right]$
Period Range	$[250, 1500]$
Phase Shift Range	$[-1, 1]$

Table A.3: Compatibility Distance Parameters.

Parameter Name	**Value**
Disjoint Coefficient	2.0
Excess Coefficient	2.0
Weight Difference Coefficient	1.0
Fitness Coefficient	0.0

Table A.4: Speciation Parameters.

Parameter Name	**Value**
Compatibility Threshold	6.0
Compatibility Modifier	0.3
Species Size Target	8
Dropoff Age	15
Age Significance	8.0
Survival Threshold	0.2
Smallest Species Size With Elitism	5
Mutate Species Champion Probability	0.0

Table A.5: Multi-Objective Parameters.

Parameter Name	**Value**
Include Genotypic Diversity Objective	Yes
Genotypic Diversity k	15

Table A.6: ODE Parameters.

Parameter Name	**Value**
Step Size	0.001s
Evaluation Length	12500 time steps
Contact Surface μ (ice)	0
Contact Surface μ (ground)	*dInfinity*
Contact Surface Slip1	0.01
Contact Surface Slip2	0.01
Contact Surface Soft ERP	0.96
Contact Surface Soft CMF	0.01
Contact Max Correcting Velocity	0.01
Contact Surface Layer	0.001
Motor FMax	0.5
Motor K_p	3
Motor K_d	0
Motor K_i	0
Gravity Vector	$(0, 0, -9.8)$
Linear Damping	0.005
Angular Damping	0.005
Trimesh Density	0.1
Capsule Density	5.0

Appendix B: Reproduction Depth-Controlled Shadow Models

As described in the text, the naïve shadow model (random selection) does not provide a good control case for investigating the changes in morphological complexity over time because at a given generation individuals in the random selection experiments will be the end product of many more reproduction (mutation and crossover) events than the corresponding individuals evolving for displacement. This is because the majority of mutations are detrimental for a real fitness objective, and so it is likely that newly introduced individuals will be discarded in the displacement experiments, while under random selection they are just as likely to reproduce as any other individual. Therefore, these individuals will have had many more opportunities to add to the complexity of their genomes and hence the complexity of their morphologies. In order to create a neutral shadow model that does not suffer from this bias, alternative reproduction depth controlled shadow models are created.

For each independent trial of CPPN-NEAT for which shadow models will be run a record of every reproduction event is created as follows: When going from generation t to generation $t+1$ there are survivors (S), mutants (M), and individuals resulting from crossover (C). For each instance of each of these classes the number of mutation and crossover events in the ancestral line leading to that individual is recorded. So for an individual $s \in S$, whose evolutionary history contains i mutations and j crossovers, an entry (i, j) is added to the list of survivors at generation t and s maintains that it had i mutations and j crossovers. Similarly if individual m produces mutated offspring m' (either by adding additional nodes or links, or by altering existing genetic material), and in the evolutionary history of m there were i mutations and j crossovers, an entry (i, j) is added to the list of mutants at generation t and m' then gets recorded as having $i' = i + 1$

mutations and $j' = j$ crossovers. Finally if c_1 and c_2 with histories (i_1, j_1) and (i_2, j_2) cross to form c', an entry $((i_1, j_1), (i_2, j_2))$ is added to the list of crossovers at generation t and c' gets recorded as having $i' = \max(i_1, i_2)$ mutations and $j' = \max(j_1, j_2) + 1$ crossovers.

For each trial of CPPN-NEAT, a reproduction depth controlled shadow model is run whereby at each generation individuals that match the reproduction profiles of the trial being shadowed are randomly selected. For example if at generation t in the real trial there were k survivors with evolutionary histories $(i_1, j_1), (i_2, j_2), ...(i_k, j_k)$ then in the shadow model there will also be k survivors, the first of which is randomly selected from all individuals in the current population which are the result of i_1 mutations and j_1 crossovers, the second of which is randomly selected from all individuals in the current population which are the result of (i_2, j_2) reproduction events, and so on. The same is done for the mutation and the crossover events. Running a shadow model in this fashion guarantees that each real trial will have a shadow where selection is random, but at each generation there are individuals with matching numbers of mutation and crossover events.

There are a few additional caveats here. First, in order to compute morphological complexities, valid morphologies are needed[1]. So, if there are individuals matching the different reproduction events that also produce valid morphologies, those are given preference over those individuals that do not produce valid morphologies (similarly to what is done for completely random selection). Second, while there will always be individuals in the population matching the appropriate histories (since they are being followed from the outset) the speciation in CPPN-NEAT is dynamic and so there might not be individuals who match the crossover profiles and are in the same species. Crossing over between species is not something that would ever happen in an actual trial of CPPN-NEAT so this may be problematic. To compensate for this two different shadow models are run for each actual trial of CPPN-NEAT.

In shadow model a, for each crossover event the reproduction profiles are matched exactly with preference given to matching individuals within the same species, but if there are no matching individuals that are also in the same species crosses between individuls from different species is used as a fall back. In shadow model b inter-species crosses are never allowed (so its behavior is more in line with CPPN-NEAT), but this means it may be necessary to allow some flexibility in matching the reproduction profiles. Here, the reproduction profiles of crossovers are matched exactly if possible (i.e. appropriate individuals within the same

[1]The CPPN must output values above the threshold at some subset of the queried locations

species exist), but otherwise progressively larger deviations from the reproduction profiles are allowed until an appropriate intra-species crossover is found. These deviations can cascade though and so it is necessary to follow the same procedure of allowing progressively larger deviations in reproduction profiles for survivors and mutations as well. Shadow model b eliminates any bias that might be introduced from allowing inter-species crosses, but may introduce its own biases by not matching the reproduction profiles exactly. However, both models have similar complexity curves (see Fig. 9.9) indicating that neither bias has a large effect, and that this shadow procedure is robust to whichever alternative is employed.